Inside This Issue

Enterprise Database Technology Pg. 14

Virtual Private Network's Pg. 34

Cron & Crontab for SysAdmins Pg. 68

MacTech Magazine
March • 2005

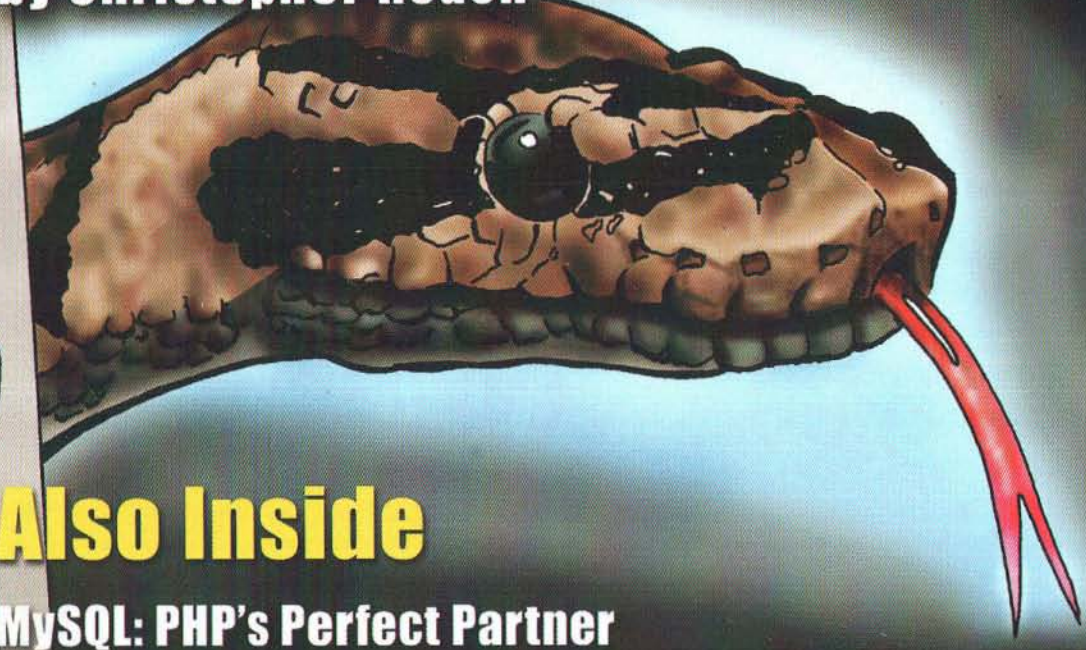
MACTECH®

The Journal of Macintosh Technology and Development

Programming Python

A Gentle Introduction to the Python Programming Language

by Christopher Roach



Also Inside

MySQL: PHP's Perfect Partner

Performing Basic Image Manipulation

Recommending QuickTime Programming Books

The Shell: What?

A Smart World After All

Movable Type On Panther

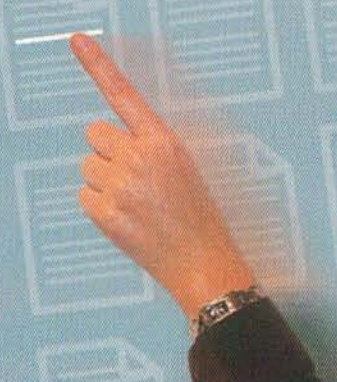
$8.95 US
$12.95 Canada
ISSN 1067-8360
Printed in U.S.A.

Need to find something fast?

INDEX INDEX INDEX

With c-tree Speed.

TRY IT NOW!



FairCom's c-tree Plus®

embedded database engine offers Superior Indexing Technology – the key to performance, data integrity, and concurrency. c-tree Plus offers direct record-oriented C and C++ APIs with an industry-standard SQL interface that allows use of any combination of APIs within the same application. Furthermore, we offer source code access for intimate programming control, unmatched portability, and developer-to-developer technical support.

Heterogeneous Environments

c-tree Plus and c-treeSQL™ Servers are the perfect solution for your mixed platform environments. Mac servers to Windows clients? No problem. Linux Servers to Mac clients? No problem. c-tree has a long history of cross-platform development solutions. Byte incompatibility between platforms is handled seamlessly with our Uniformat data handling technology.

Low TCO

c-tree is priced affordably, requires minimal hardware resources, and needs no IT staff for maintenance. If Total Cost of Ownership (TCO) is important to you, c-tree is the perfect database.

Easy Deployment

c-tree Servers are designed for ease of use and deployment as well. Out of the box, our servers can be installed and running in minutes.

Start indexing your data today!

FairCom®

USA • Europe • Japan • Brazil

www.faircom.com

Go to www.faircom.com/go/mteval for a FREE evaluation of c-tree Plus!

An Event That Changed Our Lives

30 years ago this April, two guys who's names have become household words and one no one except his mother and father has ever heard of got together and started a venture that changed the world lived in by everyone who reads and works on this magazine. The three guys? Steve Wozniak, Steve Jobs, and Ron Wayne. The Steves, with very different personalities, priorities and objectives have, by anyone's standards, achieved astounding success. As for Ron Wayne, who six days after Apple took an order for its first 50 "computers", sold his 10% interest in Apple for $800, it seems Mr. Wayne has no regrets, has moved on with his life and is happy that he made the right the decision based on the information available to him at the time. Given his original stake in Apple, if held until Apple's peak share price in 2000, would have been worth over $500 million, we have to say Ron Wayne is a man made of much stronger stuff than we!

Three recent books, in very different ways and with very different areas of focus, have done a great job of capturing the spirit of the company; its technology and the culture that has evolved around it over these 30 years. The books are:

Revolution in The Valley: The Insanely Great Story of How the Mac Was Made
By Andy Hertzfeld with a Forward by Steve Wozniak
O'Reilly Media 2004
http://www.oreilly.com/catalog/revolution/

Apple Confidential 2.0: The Definitive History of The World's Most Colorful Company
By Own W. Lonzmayer
No Starch Press 2004
http://www.nostarch.com/frameset.php?startat=apple2

The Cult of Mac
By Leander Kahney
No Starch Press 2004
http://www.nostarch.com/frameset.php?startat=apple2

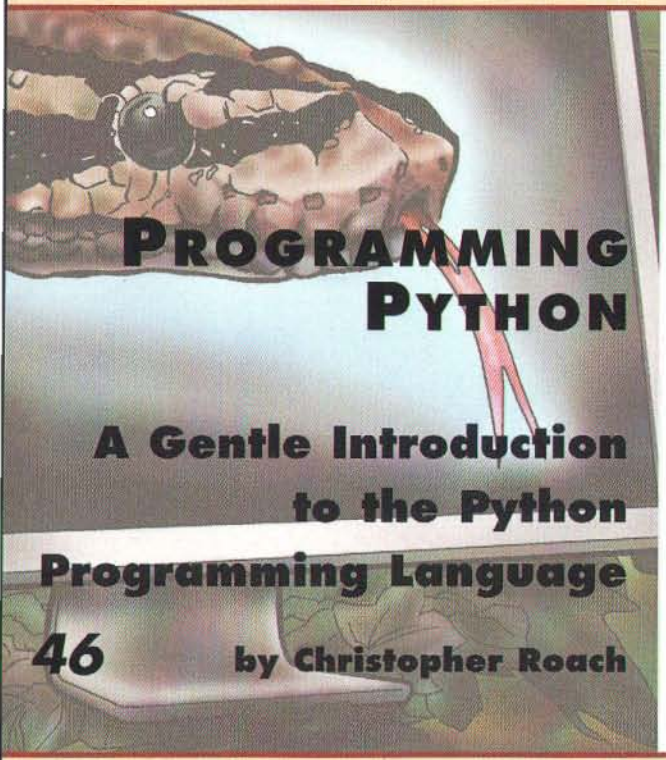
Revolution in The Valley

Andy Hertzfeld, who began his journey documenting his story of co-parenting the Mac on his site www.folklore.org, has done a magnificent job of sharing his most insiders' view of the process that went into creating the Mac. In a series of stories that feel almost like extemporaneous narrative, Andy relates the tale of the Mac's creation with such intimacy, spontaneity and spirit, the reader feels like he is right there along side Andy and his mates, sharing

MACTECH
A publication of
XPLAIN CORPORATION

TABLE OF CONTENTS

COVER STORY



FEATURED ARTICLES

DEPARTMENTS

KOOL TOOLS & REVIEWS

with them their daily quest for engineering perfection. Andy's masterful storytelling makes the reader feel more like a participant than an observer.

Highlights for us are:

- Reading Andy's handwritten technical notes and comments on A-traps
- Watching him puzzle through the process of fitting all that stuff into such a tiny memory footprint
- Seeing Bill Atkinson's pictures of the evolution of the graphics capabilities of both the Lisa and the Mac. Fascinating Stuff!!
- Andy's impressions of the Mac team, both famous and unheralded, the interrelationships of the players, learning his impressions of who were the real players vs. the imitators, who were the people who were truly on a quest to change the world as opposed to those who were simply looking for yet another way to gratify their personal ego needs. He really didn't hold anything back. When you read his thoughts in this regard you know you're getting his true feelings.
- All those old icons and screenshots of early Mac DAs, Control Panels and the Finder. Wow did that bring back memories!

This is a book for geeks! There is a lot of low level stuff on the creation of both the hardware and software, the challenges the engineers on both sides were confronted with, the trade-offs that had to be made as a result of both memory limitations and the cost points the team thought they needed to hit to ensure the Mac truly would be "the computer for the rest of us"! The folks at O'Reilly have done a wonderful job of laying out the book in an interesting yet pleasing fashion. If you love the Mac and want to hear the story of its creation from one of its principal creators, a guy who was there almost from the project's first day right up until the machine shipped, we highly recommend this book!

Apple Confidential 2.0

To be honest, when our friend Patricia Witkin, the Communications Manager at No Starch Press, www.nostarch.com, sent us a copy of Owen Lonzmayer's updated version of "Apple Confidential", our first thought was, "Just what the world needs: Another retelling of one of the most often told stories in the history of American business." You have to understand, while we have great respect for both Owen's writing ability and his knowledge of Apple and its industry, we have read virtually every word ever published on the subject. We're happy to report our first reaction could not have been more off the mark. Owen has done a really impressive job of bringing a fresh perspective to the subject. He has done a great deal of in-depth research, chose to highlight aspects of the story that are truly worth retelling, and did it all with an entertaining narrative style that drives the reader to keep turning the pages. Through access to many of the principals, he has even managed to dig-up some stories and anecdotes we haven't heard before.

Some particularly memorable (and in some cases new) stuff:

- Woz's story of how Apple got its name. Steve Jobs came up with it, from where Woz knows not; Woz initially hated it, but couldn't come up with anything they liked better. So it was Apple by elimination!

MacTech Magazine (ISSN: 1067-8360 / USPS: 010-227) is published monthly by Xplain Corporation, 850-P Hampshire Road, Westlake Village, CA 91361-2800. Voice: 805/494-9797, FAX: 805/494-9798. Domestic subscription rates are $47.00 per year. Canadian subscriptions are $59.00 per year. All other international subscriptions are $97.00 per year. Domestic source code disk subscriptions are $77 per year. All international disk subscriptions are $97.00 a year. Please remit in U.S. funds only. Periodical postage is paid at Thousand Oaks, CA and at additional mailing office.

POSTMASTER: Send address changes to **MacTech Magazine**, P.O. Box 5200, Westlake Village, CA 91359-5200.

From the Editors... Continued

- The explanation for Ron Wayne's early bailout from Apple
- The chapter entitled "Woz's Wonderings", which chronicles the adventures Woz had during his early post-Apple days. Whether it was Owen's intention or not, this chapter also captures the essence of the key differences between what drives the two Steves, and why, it's our guess, they would never be the very closest of friends.
- A good telling of John Sculley's Apple tenure
- A verbatim publishing of a 1985 memo Bill Gates sent to John Sculley and Jean-Louis Gassee that, if heeded, would have lead to the Mac OS becoming the standard desktop platform instead of Windows.
- A really good telling and analysis of the Mac clone fiasco. Anyone out there still have a Power Computer? J
- A well written and concise wrap-up chapter on Steve Job's return to Apple and the company's transformation from an IT to a consumer electronics company

Once again, we are happy to be able to strongly recommend "Apple Confidential 2.0" to anyone who is interested in the Mac, Apple, the computer industry or who just enjoys a compelling story told very well.

The Cult of Mac

It's rare in our experience when we feel comfortable using the words "unique" or "original" to describe something that's been published about the Mac, but these words apply very well to Leander Kahney's amusing and pleasurable book. Beautifully done from a design and production standpoint, "The Cult of Mac" does a great job of chronicling why people merely USE Windows machines, but LOVE Macs.

From the books cover of a guy with Apple's logo expertly shaved into the back of his head, all the way through to picture of a mom, very much in labor, with her iPod in hand, earbuds plugged firmly into her ears, Leander, who in case you didn't know is Wired News' Mac guy, does a simply wonderful job of capturing people's fascination with, deep passion for, and in many cases obsession with all things Apple.

This is a truly a visually striking book. Leander and the folks at No Starch have produced a book that would be a treat even without the wonderful narrative that accompanies the eye-catching pictures and stunning design. This book was made by people who posses the same sense of style and design that differentiate Apple's products from the rest of the marketplace. And it's written and published for the people who understand this difference and because they do, buy and love the products Apple makes.

For the record, Patricia Witkin was kind enough to make both Owen and Leander available to us for interview and comment. We wish our editorial schedule (and VERY full plates) had allowed for the time to do this. We're sure they would have had a bunch of very interesting stuff to share with our readers!

MACTECH®

Communicate With Us

DEPARTMENT E-mails

Orders, Circulation, & Customer Service
cust_service@mactech.com

Press Releases
press_releases@mactech.com

Ad Sales
adsales@mactech.com

Editorial
editorial@mactech.com

Online Support
online@mactech.com

Accounting
accounting@mactech.com

Marketing
marketing@mactech.com

General
info@mactech.com

Web Site
http://www.mactech.com

In this electronic age, the art of communication has become both easier and more complicated. Is it any surprise that we prefer **e-mail?**

If you have any questions, feel free to call us at 805/494-9797 or fax us at 805/494-9798.

If you would like a subscription or need customer service, feel free to contact MacTech Magazine Customer Service at 877-MACTECH

We love to hear from you! Please feel free to contact us with any suggestions or questions at any time.

Write to letters@mactech.com or editorial@mactech.com as appropriate.

MySQL: PHP's Perfect Partner

I figure, if you're reading this article, you are probably pretty new to the database universe and, therefore, MySQL is a good choice for you. But just in case you've wandered into the room with some existing expertise, but are new to open source solutions like MySQL, here's a great URL that will tell you what MySQL does differently than ANSI Standard Query Language:

http://dev.mysql.com/doc/mysql/en/Differences_from_ANSI.html

MySQL is rock-solid, and incredibly fast. There are more than five million active MySQL installations in the world and MySQL has been downloaded more than ten million times. Most are of the LAMP variety (Linux/Apache/MySQL/PHP/Perl). Lots of Mac and Windows setups as well. There are a number of benchmarks that show MySQL as the fastest such systems available, faster than the most expensive commercial DBMS's, faster even than PostgreSQL.

MySQL uses the GNU General Public License (GPL). Want to run it on your personal computer? No charge! Compare that to the $1,000 per seat licenses of some commercial DBMS apps, or the $50,000+ cost for some commercial server packages. And chances are good that if your ISP offers PHP, they'll offer MySQL as well. MySQL is not hard to find.

In this month's column, we're going to install MySQL, then make sure it's set up and ready to use. If you don't already have a reasonably recent version of PHP installed on your computer, now would be an excellent time to do so.

Installing MySQL

With each new release, the folks at MySQL AB (the corporate entity that owns the rights to MySQL) have made it easier and easier to install MySQL. In the early days of Mac OS X, installation was a bear. You had to locate the source code, build and debug, searching the net to find info on the many compile switches, till you finally made your way through a successful build. You then needed to deal with ownership issues to make sure someone couldn't creep over the net and illicitly access your tables. Working with early MySQL releases required a real pioneering spirit. Nowadays, installation is fairly straightforward, almost trivial.

Early versions of Mac OS X required that you go through the process of creating a new user named mysql. That user was given ownership of the installed files that were not owned by Root. Most folks added the mysql account using the *Accounts* pane in *System Preferences*. That worked fine, though it created some files and directories that would only be used by a human user and added that user to the set of users presented at login.

Nowadays (since the release of Mac OS X 10.2), Apple takes care of this bit of business for you. Though they don't do the installation of MySQL, recent versions of Mac OS X create the mysql user as part of the System install. To see this for yourself, go into your Applications directory, Utilities subdirectory, and launch *NetInfo Manager* (in the Finder, note that shift-command-U is a shortcut to the *Utilities* directory).

When NetInfo Manager's main window appears, use its browser to locate the *users* directory. You should see a user named *mysql* in the second column (see Figure 1). Notice the values for *home* and *shell*. Since we won't be logging in as mysql, there's no reason for a shell and home directory.

A few month's back, I wrote about PHP. We walked through the installation process, then went through the basics. PHP is a wonderful tool, all on its lonesome. But boy does it shine when you add a database backend to the mix. There are several database systems that work well with PHP. If you're relatively new to this business, or if your database requirements are not particularly sophisticated, MySQL is the perfect choice.

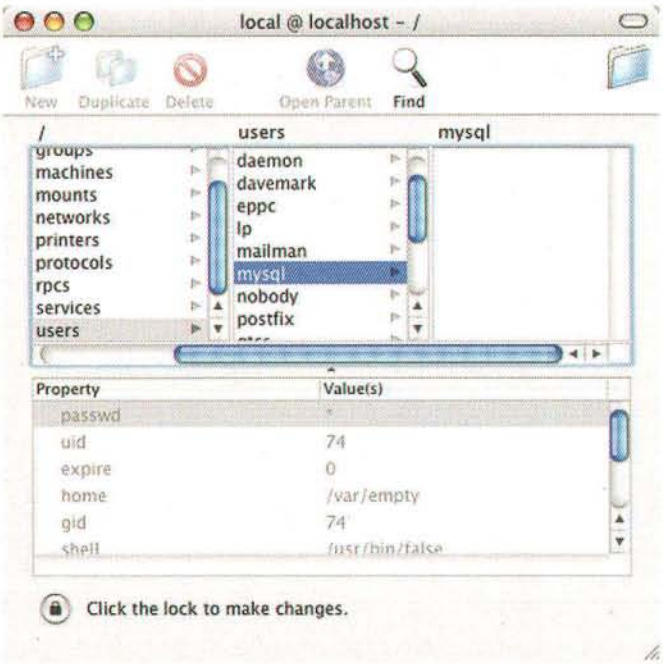


Figure 1. NetInfo Manager, showing the mysql user.

Downloading MySQL

Though the net is full of tons of excellent MySQL resources, by far the most important is found at MySQL's official home at http://www.mysql.com. Most of the stuff we'll be interested in lies behind the *Developer Zone* tab at http://dev.mysql.com. To start your download decision-making process, navigate to the main download page:

http://dev.mysql.com/downloads/

Figure 2 shows the links of interest when I navigated there. Notice the *Mirrors* link. When you click to this page, the site will use your IP address to build a list of mirror sites it thinks are geographically close to you. You'll definitely want to check this page out if you run into problems downloading from the main site.

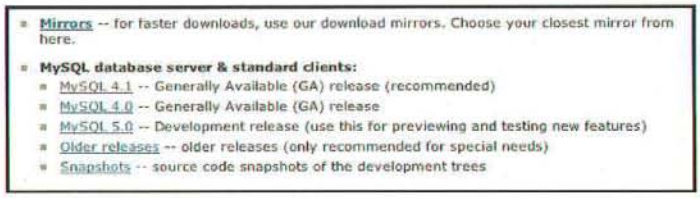


Figure 2. The important links on the downloads page.

Next on the list is the latest GA (Generally Available) release, followed by the previous GA release. After that is the latest preview release (essentially a beta). I would definitely stick with the most recent GA release. Click on that link.

For me, the most recent GA release was MySQL 4.1. Clicking on that link brought me to the *MySQL 4.1 Downloads* page. Take a minute to read the text at the top of the page:

> The MySQL database server is available under the MySQL AB "dual licensing" model. Under this model, users may choose to use MySQL products under the free software/open source GNU General Public License (commonly known as the "GPL") or under a commercial license.

Click on the GNU General Public License link, then on the commercial license link. In effect, you can use the MySQL server at no cost to you, if your app is 100% GPL. This is the beauty of Open Source and the GPL model. If you are not

familiar with GPL or are new to Open Source, it is well worth your time to read through the MySQL license pages. I think they are very well written and very understandable. Worth taking the time to do this.

If you are just in learning mode, the GPL license is fine. The commercial license is for people who do not want to release their source code or who find the GPL licensing rules too restrictive. The cool thing is, even the commercial license is relatively inexpensive when compared to other commercial products.

Scroll down the *MySQL 4.1 Downloads* page until you come to an area labeled *Mac OS X downloads*. As you can see in Figure 3, this part of the page is divided into 4 different sets. Two are installer-based, two are tar-ball based. There's one of each type for Jaguar and one for Panther. We're going for the Panther version with the installer package.

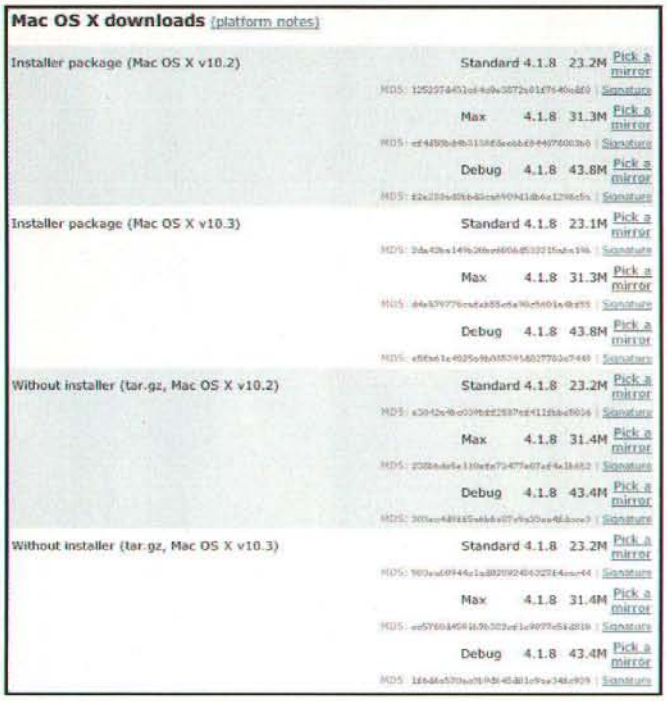


Figure 3. The Mac OS X download options.

Now that we've decided that, we need to decide between *standard*, *max*, or *debug* versions of the server software. You'd use the *debug* version if you were trying to debug the MySQL source itself, or if you were trying to track down a particularly knotty problem in your code and needed to see the MySQL symbols. Note that, as with any software package, there's a significant performance hit associated with the debug version.

The *max* version includes a number of esoteric features that you most likely won't need (the NDB storage engine, Berkeley DB storage engine, UDFs, BIG_TABLE support, etc.) The features in *max* tend to be beta in nature and will migrate to the *standard* release as they stabilize.

Bottom line, *standard* is the one you want. So, if you have Panther installed, you'd go to the *Mac OS X downloads* section, then click on the 4th overall *Pick a mirror* link. It'll be the first link in the subsection labeled *Installer package (Mac OS X v10.3)*.

Once the mirror page appears, you might want to use the login link to create a new MySQL account and login. With a login, you'll be able to post questions to the forum, subscribe to the MySQL newsletter, etc. Worth it.

Once you're logged in, click on the closest mirror, then go get a nice piece of halvah. Tap, tap, tap. Done yet? Ah, there you go. If you downloaded the installer package, you'll get a *.dmg* file which should automatically mount as a disk image. Open the image. You'll see two packages and a readme file. The first package is the MySQL server package. The second installer, called *MySQLStartupItem.pkg*, installs a Startup Item which will automatically start up the MySQL server at boot time. If you are going to spend any amount of time with MySQL, you'll want this Startup Item installed.

Start with the main package. Then install the Startup Item. If you run into any problems, dig into the readme file. There's a *lot* of helpful info in there.

Starting the Server for the First Time

Your next step is to start the MySQL server, so we can start to play! The simplest way to do this is to just restart your computer and let the Startup Item do its thing. But it's worth seeing how this is done by hand, just to get a sense of how this works.

Fire up Terminal, then type this command:

```
man -ps
```

When you hit return at the end of the command, one page worth of the manual page for the *ps* command will display in the Terminal window, and a colon (:) prompt will appear at the bottom of the screen. You are looking for the list of options to the ps command, specifically descriptions of the "-a" and "-x" options. To move down a page, hit the space bar. To quit, either hit enough spaces to scroll to the end or type the letter q.

The listing for "-a" says, "Display information about other users' processes as well as your own." The listing for "-x" says, "Display information about processes without controlling terminals." Let's combine these two, like so:

```
ps -ax
```

You'll see a long scrolling list of processes. At this point, none of them should have the word mysql in them. Unless you have a really wide monitor, the commands will likely get clipped, making them hard to read. Try this command instead:

```
ps -ax > textfile
```

This does the same thing, but redirects the process listing into a text file named *textfile*. Unless you've specifically changed directories since you started up Terminal, the file should be in your home directory. Go into the Finder, look in your home folder, and drag *textfile* onto TextEdit. That's better!

Now let's start the server. At the command prompt, type:

```
sudo /Library/StartupItems/MySQLCOM/MySQLCOM start
```

The sudo command is asking Unix to do this command as super user, or root. You should be prompted for your root password. Once you successfully enter your password, you should see this message:

```
Starting MySQL database server
```

Cool. Now do your ps -ax again and you should see two process entries that resemble these two:

```
  388  ??  S      0:00.03 sh ./bin/mysqld_safe -
datadir=/usr/local/mysql/data -pid-
file=/usr/local/mysql/data/Dave-Marks-Computer.local.pid
  408  ??  S      0:02.06 /usr/local/mysql/bin/mysqld -
defaults-extra-file=/usr/local/mysql/data/my.cnf -
basedir=/usr/local/mysql -datadir=/usr/local/mysql/data -
user=mysql -pid-file=/usr/local/mysql/d
```

The first entry is the shell wrapper for the server daemon. Basically, this shell is acting purely as a wrapper and a safe way to communicate with the server.

The second entry is the server daemon itself. Though the daemon does all the work, you shouldn't have a need to communicate with it directly. Notice that the daemon is running with *user=mysql*, and not as root. This is the right way to do this. Running as root would create a dangerous security hole.

Want to shut down the MySQL server? Don't worry, it's perfectly fine to do this. Type this command:

```
sudo /Library/StartupItems/MySQLCOM/MySQLCOM stop
```

Check your ps -ax again. The two processes should be gone. Go ahead and start the server again, so we can play a bit. When you restart your machine, the Startup Item issues the same command you're using to start the server:

```
sudo /Library/StartupItems/MySQLCOM/MySQLCOM start
```

Setting Up the Aliases

To help save some typing, let's set up a couple of aliases. If you are using *bash* shell, type these two commands:

```
alias mysql=/usr/local/mysql/bin/mysql
alias mysqladmin=/usr/local/mysql/bin/mysqladmin
```

If you are using almost any other shell, type these two commands:

```
alias mysql /usr/local/mysql/bin/mysql
alias mysqladmin /usr/local/mysql/bin/mysqladmin
```

If you are not sure which shell you are using, check the title of the Terminal window. It should say. Or just type one of the sets above. If you get an error, try the other set.

Once you've successfully executed one set or the other, add the two lines to your shell's startup file so these two aliases will be setup automatically each time you open a new Terminal window. For now, just type the commands and leave the Terminal window open so the aliases stick around.

Now, if you type *mysql*, you'll execute the command */usr/local/mysql/bin/mysql* and when you type *mysqladmin*, you'll execute the command */usr/local/mysql/bin/mysqladmin*. Aliases are *very* useful.

Setting Up the MySQL Accounts

Our last step before we actually start playing with MySQL itself is to secure the default MySQL accounts and set up a non-root account for our dabbling pleasure.

MySQL ships with two root accounts and two anonymous accounts that do not have passwords. Obviously, a dangerous situation, though one that makes perfect sense from the vendor's perspective.

There are a number of ways to do this. We'll use the *mysql* alias we just set up. In Terminal, type this command:

```
mysql -u root
```

This command starts up the *mysql* monitor using the *root* user. Note that this is not the same as your Unix *root* account. MySQL maintains its own list of users, as well as its own data security model that allows these users to own the MySQL data. Normally, when you start up the *mysql* client, you'd type a user name *and* a password. Since there is no *root* password yet, all we need do is specify the user name. *mysql* will reply as follows:

```
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1 to server version: 4.1.8-
standard

Type 'help;' or '\h' for help. Type '\c' to clear the
buffer.

mysql>
```

Notice that you are now running the MySQL monitor. The prompt at the bottom of the Terminal window is the standard MySQL prompt. You can exit the monitor by typing the command *exit*, followed by a return. Don't do this quite yet!

MySQL ships with a user table that holds all its account info. Let's ask the monitor to list the host and user columns in that table. At the *mysql>* prompt, type this command:

```
select host,user from mysql.user;
```

Notice the semicolon (;) at the end of the command. Very important!!! The semi tells the monitor that we've reached the end of the command. Here's the results on my computer:

```
+-------------------------+------+
| host                    | user |
+-------------------------+------+
| Dave-Marks-Computer.local |      |
| Dave-Marks-Computer.local | root |
| localhost               |      |
| localhost               | root |
+-------------------------+------+
4 rows in set (0.64 sec)

mysql>
```

Notice that I've got 4 accounts. Two root accounts, two anonymous accounts. One of each type is for connecting from

the local host. The other is for connecting from any other host.

Let's add some passwords to these accounts, keep the bad guys out! Still in the monitor, type this command, replacing *xxxxx* with the password you want for your local anonymous account, *yyyyy* with the host name from the host column above (the entry in that column that is *not* localhost) and zzzzz with the password you want for your second anonymous account:

```
SET PASSWORD FOR ''@'localhost' = PASSWORD('xxxxx');
SET PASSWORD FOR ''@'yyyyy' = PASSWORD('zzzzz');
```

This is the reply I got to each of these commands:

```
Query OK, 0 rows affected (0.00 sec)
```

If you'd like some evidence that you just changed the password, try this command:

```
select password,user from mysql.user;
```

Here's the result I got, after I added my passwords:

```
+-------------------------------------------+------+
| password                                  | user |
+-------------------------------------------+------+
|                                           | root |
|                                           | root |
| *18796D3E621A0FB8F69503C1006CF26D337330   |      |
| *18796D3E621A0FB8F69503C1006CF26D337330   |      |
+-------------------------------------------+------+
4 rows in set (0.00 sec)
```

Notice that the password column is stored in an encrypted form, as you might expect. Notice also that the root passwords have not been set yet. We'll do those next. Finally, note that I changed both my passwords to the same value. Good strategy? Perhaps not, but I wanted to show the consistency of the encryption. In real life, I delete the anonymous accounts completely, since I don't like them hanging around. Want to try this? Here's how you delete your anonymous accounts. Do *not* type these commands, unless you really don't want your anonymous accounts!!

```
DELETE FROM mysql.user WHERE User = '';
FLUSH PRIVILEGES;
```

The first command deletes the unnamed users from the user table. The second command is necessary since the table is only read when the server is first started. This prevents us having to restart the server.

Our last task is to set passwords for the root accounts. As you did before, substitute your new root password for *xxxxx*, your host name for *yyyyy*, and your second root password for *zzzzz*.

```
SET PASSWORD FOR 'root'@'localhost' = PASSWORD('xxxxx');
SET PASSWORD FOR 'root'@'yyyyy' = PASSWORD('zzzzz');
```

Feel free to use this command again, to check your results:

```
select password,user from mysql.user;
```

Here's my results:

```
+-------------------------------------------+------+
| password                                  | user |
+-------------------------------------------+------+
| *18796D3E621A0FB8F69503C1006CF26D337330   | root |
| *18796D3E621A0FB8F69503C1006CF26D337330   | root |
+-------------------------------------------+------+
2 rows in set (0.00 sec)
```

Notice that my anonymous accounts are gone. If you somehow forget your root password after you do this, here's a link to a page that tells you how to reset the password:

http://dev.mysql.com/doc/mysql/en/Resetting_permissions.html

Till Next Month...

We did a *lot* this month. But the real fun comes in my next MySQL column when we really get into this stuff. We'll create tables, add data and, eventually, use PHP to pull that data out of the database and display it in a web page. Cool!

Not sure if we'll do all this next month, but I'll try. In the meantime, be sure to check out the new books at http://spiderworks.com. Rumor has it that there's a series of Tiger books in the works. Automator, Dashboard, and Spotlight. Excellent! See you next month... ☺

MT

About The Author

Dave Mark is a long-time Mac developer and author and has written a number of books on Macintosh development. Dave has been writing for MacTech since its birth! Be sure to check out the new Learn C on the Macintosh, Mac OS X Edition at http://www.spiderworks.com.

The new **HASP** family of products is the next generation in protection ensuring the highest level of security for your software. It provides an easy to use set of tools to protect once and deliver through a variety of licensing options.

1 Develop
- For Windows, Mac OS, Linux

2 Protect Once
- Data encryption with industry standard hardware-based AES 128-bit
- Strong wrapper with code obfuscation prevents debugging and reverse engineering
- API code generator for easy implementation and customization

3 Distribute Many
- Single and multi-user license management
- Secure license updates for keys in the field
- Innovative licensing models defined separately from protection

Get the kit: Developer's Software, Guide and Demo Key at www.aladdin.com/hasp

Aladdin
SECURING THE GLOBAL VILLAGE

North America: 1-800-562-2543 **International:** +972-3-636-2222 **UK:** +44-1753-622266 **Germany:** +49-89-89-42-21-0 **Benelux:** +31-30-688-0800 **France:** +33-1-41-37-70-30 **Spain:** +34-91-375-99-00
Israel: +972-3-636-2222 **Asia Pacific:** +852-21-66-8605 **Japan:** +81-426-60-7191

Enterprise Information Systems on Mac OS X

Confronting the Myths and Challenges of the Enterprise World

By David Swain

There was great anticipation in the capital city of the Land of Smiles! Among the young, who had never seen other times, there was talk of a new Golden Age coming where their new data engines would once again help their land become a world power in commerce and an even mightier force in the arts. They were rightfully proud of the achievements of their land's techno-artists and of the world's renewed interest in their advanced electronic creations.

But those who were older and who did remember the earlier Golden Age viewed the coming boom in commerce with both joy and dread. Yes, it was possible that a much larger share of the world's wealth of information would now rely on their people and technology for safekeeping and secure access. And it was possible that great economic rewards would enrich their fair land as a result of this newfound global respect. But it was also possible that a few key mistakes made at just the wrong moment and in just the wrong place could bring disaster where there should have been success. This could build an even greater barrier between the Land of Smiles and the outside world, forever closing them off from the great spiritual and physical treasures that could have been theirs. These elders remembered how the Land of Smiles really fell out of favor with the business world years ago...

Myth vs Reality

There are a number of myths that circulate through the enterprise world and shape the behavior and decisions of those who work in that environment. Myths are very powerful when enough people believe them. They are cautionary tales intended to keep the inexperienced from harm. But if they are still believed by those who have otherwise reached maturity, they can impede progress and stifle growth.

As with any myth, enterprise information technology myths may very well have some basis in fact or in history. But they are more often misinterpretations, sometimes deliberate, of facts or events that are fueled by fear of the unknown and fanned by people with something to sell. Even if the basis of a myth was true at some historical time, it may no longer be true. Continuing to foster such a myth could become a barrier to progress.

Such is our first myth...

Myth #1 – There Are No Enterprise Database Tools for Macintosh

Not all in the Land of Smiles had abandoned the rituals of data management with the native technology in those years. But those who persisted in this practice often did so in secret, fearing ridicule or banishment. Many sought refuge in other

lands in order to ply their craft - and many of those became swallowed up in the foreign culture that had become their home. Some even began to believe the vicious rumors they heard from their neighbors about the Land of Smiles, or at least they nodded in assent in order to blend in. But most longed in private for the day they could return to their beloved land and freely pursue their chosen trade...

Of the widely held beliefs in the enterprise database market, one of the most prevalent categories of belief regards the use of Apple products and Mac-based software for "serious" work. Basically put, the core belief is that "there are no enterprise-level database application development tools" built for this platform. With the growing number of powerful Apple server products being purchased, with the return of more "visible" database products like Oracle to the Macintosh platform, and with a little education, this misperception can change.

But to facilitate this change, many of us will have to learn some new skills as well. And we must also learn to use the ***right*** tools to do the best job. If we choose the simpler tools with which we are familiar in the small business market to attempt to build mission-critical applications for the enterprise market, we may find ourselves becoming the source of ***new*** myths that the Macintosh platform is not suitable for professional database work. It is not a matter of whether we ***can*** use a certain tool, but whether we ***should*** use that tool when there are better tools available.

Years ago there were many ***small*** business systems built using software that ran on the Macintosh operating system – some better than others. It was a simpler time when the word "gigabyte" had never been uttered in polite conversation and the tools of the day on ***any*** platform were much less sophisticated (or capable) than they are now. Some database companies that remained with the Macintosh platform since that time have made great progress. And the increasing acceptance of Mac OS X Server and Apple server and mass storage products should bring more of the major database server companies back as well. But how well equipped for enterprise database work is Mac OS X even now?

The State of the Platform

It is not well known in enterprise database circles, but Mac OS X has ***never*** been without powerful enterprise-level tools for information management. While the coming of Oracle has brought much-deserved attention to Mac OS X and Mac OS X Server as database client and server platforms, equally capable products have been with us since the beginning. The very day Mac OS X officially shipped (Saturday, March 24, 2001), an excellent SQL database product named FrontBase officially shipped its new Mac OS X version, joining successful versions already running on Linux and Unix systems. Another fine SQL database named OpenBase also shipped for OS X at that time, having also spent the long OS X beta period honing its abilities. (We are passing over the MySQL that came installed with OS X because it was still missing some important pieces for enterprise work.) More such products have come to the party since then.

The day ***before*** OS X shipped, a database client application development tool named Omnis Studio shipped its first version designed to run on Mac OS X, filling out a product line that already ran on Classic Mac, Windows, Linux and Sun Solaris. It was already capable of creating client applications on Mac OS X for database servers running on other platforms, but from the beginning it was also able to connect with FrontBase as part of a joint effort between the two companies.

The programmers at FrontBase had created a link (called a Data Access Module, or DAM) that allowed Omnis Studio to serve as a "front end" for their product using the native syntax of FrontBase. The creators of Omnis Studio had developed this DAM technology over many years and that program shipped with a number of DAMs for other database engines. But they also make the APIs for this technology available to SQL database companies with legitimate needs so that those companies can build an Omnis Studio DAM for their own product. Doing so in essence gave FrontBase an excellent and full-featured GUI development tool for both desktop and web browser-based applications without having to build one themselves. The good folks at OpenBase followed suit by publishing their own native DAM for Omnis Studio a few months later.

The beauty of the DAM technology is that it allows Omnis Studio to speak the native SQL dialect of each database engine while maintaining a single code base in an Omnis Studio application, in essence becoming a universal translator among all the SQL engines for which an Omnis Studio DAM exists. But is this really necessary? Isn't SQL already a standard and universal language?

Myth #2 – SQL is a Standard Language

The world had become a confusing place in that dark age. Only people could make such simple things so complicated.

Everyone in the whole world spoke the same language - but not exactly. They all used the same words, but many of the most important words often had vastly different meanings from one group to the next. Words that speak of love in one society elicit great offence in others.

No one knew how this came about, but everyone agreed on one thing: Their own group's use of language followed the one true and correct tradition. All others were vulgar dialects, unfit for civil discourse and indicating a lower intellectual capacity on the part of the speaker.

No fence is as high as the one built by the mind...

SQL a universal language? Almost true... But in practical terms, SQL is a ***model*** for a database language ***based*** on a standard. Its implementation in actual database products varies widely. We cannot go out and purchase "SQL". If all SQL databases were exactly alike, there would be no reason to have more than one of them. Look at the marketing spin of a few products:

FrontBase prides itself in strictly adhering to the SQL/92 standard, for example, to the point where they even suggest that programmers purchase C. J. Date's ***A Guide to the SQL Standard*** as a reference manual to their product. But even FrontBase adds extensions to that standard for practical reasons. OpenBase published a 14-page white paper on how their product complies better with a variety of data integrity and security standards "which every database should meet, but few do" to distinguish themselves. And Oracle is, well, ***Oracle!*** ... to the point where everyone else mentions in their own campaigns how they are equal to or better than Oracle at something important, but still ***much*** more affordable.

Why emphasize this here? The point is that there are significant differences among SQL database products. On many scales, neither is better or worse than the other. Each choice has its trade-offs. ***Switching*** from one to another is where their differences ***really*** make a difference to us. Two actors can work from the same script in entirely different ways and both create equally brilliant performances. But problems would arise if we suddenly had to swap in, say, Hoffman for Pacino after all the shooting on a movie had been done (just because new management was better friends with Hoffman's agent). It doesn't matter that the script was the same for each performance; the ***interpretation*** of the script is the reality we must deal with.

Fortunately, in an enterprise database application we don't necessarily have to rewrite the whole thing just because a decision is made to change database engines. This is because of the way such applications are structured.

Division of Labor

If you have only worked with file sharing or self-contained database products, then you may not be fully aware of the division of labor in a client/server system. Briefly, there are two parts to a client/server database application: the Database and the Application. While this may not seem to be a blinding flash of brilliance or a startling revelation, it is a distinction that is often blurred in practice – to the detriment of the finished system. Here is how these two parts could be defined:

- The Database part protects and serves the information for the system. This is also called the ***back end*** of the application. It is made up of the SQL server engine and data repository that reside on a database server machine (or cluster). The SQL server sits on this machine and responds to requests for accessing the various databases it manages. Data access requests include select, insert, update and delete commands, but there are also requests for changing the structure of a database and for managing user access to a database or specific parts of it.

- The Application part includes programs running on client machines elsewhere on the network that provide the GUI interface for operations such as presenting data, accepting data entry and generating reports. This is also called the ***front end*** of the application. The Application sends requests to the Database and receives responses from it, but it also may perform other tasks and may interact with other services (such as email) and devices (such as printers).

Of course, opinions vary as to where the boundaries of these two domains should really be placed, but the division is well recognized. Some database publishers actively encourage including functional bits of application code directly in the database, partly as a means of discouraging shifting to another database in the future. Developers at some "stable" installations (where they are fairly certain management will not change databases on them) find that this practice has some performance benefits. But many application developers who must serve users of various SQL engines prefer to put as much functionality into the Application side as possible for easier portability.

Client and server programs will generally reside on different machines (except possibly for web-based application servers) – and those machines can even be running different operating systems. As long as a communication bridge can be established, it doesn't matter where the Database and Application are relative to one another. But for our purposes in this series of articles, we will focus on 100% Mac OS X solutions, with both client and server machines running on some kind of modern Macintosh machine.

The issue of management's effect on database decision-making has been broached. Let's look at this a bit more closely.

Myth #3 – Our Company Has Standardized On XYZ

Data artists were more often servants than masters. Theirs was an obscure, secretive trade with many mystical words of power to be mastered. But they had protected their craft so well that true skill was rare, so many of the biggest businesses succumbed to the temptation of out-sourcing to other lands where intellectual labor was cheaper. This, in turn, had often led to less than satisfactory results, but managers are not programmers and their expertise is in budgets rather than in bits and bytes...

And managers were also servants of even more powerful masters. When these masters were displeased, it was the managers who bore their wrath. This was both a blessing and a curse for the data artist...

For the past many years now, the enterprise IT sphere has been in flux, with management turnovers happening frequently. Changing management often signals changing database platform. Typically, when a new VP of IT enters office, an impression must be made that immediate progress arrived with the change in management. A common tactic for making a strong first impression is to change the database platform and/or operating system on which that company is "standardized". The rationale for such changes is most often that the new platform is "more powerful" or "more standard" or even "less expensive in the long term" or arguments along those

lines, but the real reason is as likely to be that the new VP of IT (and the upper management staff that arrived on those coattails) is simply more familiar with that product or has some relationship with that company.

Sometimes it doesn't even require a change in management to shake things up. Service contracts come up for renewal periodically and computers bought in a bundle come to "end of life" and "need" to be replaced in a single, massive move. Negotiations over price of updating can be enough to cause a change of "standards".

None of this is necessarily bad, as this tactic may bring more corporations to Mac OS X sooner. And being in bed with a major database supplier – at least for the span of a few years – may actually have benefits for a corporation. But however it occurs, such a change causes an immediate major headache for the staff in the trenches, who must actually work with the new hardware and/or software and solve the problems brought about by management's change of heart. What problems? The problem that both the data and the application from the current system must be converted to the new environment.

Why is this a problem? Well, SQL comes in as many flavors as there are vendors with SQL databases as we have already seen. Also, different operating systems require different software – and some database products are only available for certain platforms. If management is fully committed to a proposed change in standards (that is, if they ***insist*** that there be ***no*** exceptions), then a number of changes could cascade from a single decision.

For example, a company that had been running MS SQL Server and that now intends to move to an all Macintosh network must also decide on a new database vendor, since Microsoft does not make a Mac OS X version of that product. Sure, this company could keep a lonely outcast Windows machine on the network and continue to use it as a database server, but that goes against their new "standard". If the Application is also written using some tool that does not exist for Mac OS X (like Visual Basic), then that has to be entirely rebuilt as well. Many believe that job security is a wonderful thing, but the unreasonably short timetables that sometimes come with such turnovers could even shake the foundations of ***that*** belief in the short term.

A parallel problem confronts those who would like to develop vertical market applications for selling into some segment of the enterprise market. Many corporations are adamant about ***only*** using their (current) "standard" database platform – and this "standard" varies from one company to another (as well as from year to year in some cases). This makes the developer's job more difficult if their code needs to be built for each specific back end. Supporting multiple code bases is a thankless task and is a real pain when it comes to providing bug fixes and upgrades!

I hear someone in the back mumbling something about ODBC. True, many database products can be accessed using ODBC (acronym for Open DataBase Connectivity), but this does not yield the same kind of performance and access to security features as does using the native dialect of each specific database product. ODBC is merely a technology, like SQL, that was originally intended to give applications (such as spreadsheet and chart generation programs) that live outside the database application access to that data for specialized tasks. Such access would generally only be one way – ***reading*** data but not updating it. Just like Omnis Studio DAMs, ODBC drivers must still be written for each database platform we need to access.

Certainly ODBC is the native dialect of MS SQL Server (after all, ODBC began with Microsoft) and a few other products, but it does not by itself give access to important (and sometimes proprietary) internal features of specific SQL products. ODBC compliance is still an important feature for applications that need to query SQL databases, but it is not part of the SQL standard.

JDBC (the Java cousin of ODBC from Sun) is a Java API for connecting to SQL databases and "other tabular data sources". It has similar intent and limitations to those of ODBC, but offers its API to the Java language rather than to variants of C.

But what if we want a ***complete*** database application development platform – one that can take advantage of built-in ***and*** proprietary features for a variety of SQL products, as well as the more common and mundane tasks, as well as provide a rich GUI interface all within a single code base? What if we need to be prepared for occasional, or even frequent, transfers of data between different SQL products? Where might we look for a programming platform that can serve these needs?

Omnis Studio to the Rescue

When she was younger, in the previous age, she was very serious about protecting and retrieving data and about interacting with the largest and most powerful engines - even the foreign ones. But she did not smile well. This did not bring her favor in the Land of Smiles, so when others appeared who had far better smiles, her countrymen turned their backs on her to follow them. But too late they learned that smiles are not all that is important in this work and so many projects failed that the Land of Smiles had to bear the shame for many years.

She had spent the last age in the shadows, gaining knowledge and power, always seeking to fulfill her purpose in facilitating data access. She journeyed to the other nations - the Land of the Sun, the Land of the Penguin and the Land of a Thousand Flags - and had dwelt among their data masters and had learned the special ways of each. She speaks the languages of all of their data engines as though she were a native... and she has learned the proper smile for each land - even of the land where smiles are rare.

Now the land of her birth is again ready for greatness in the information world and she is ready to lead the transformation! She is Omnis, the communicator, the facilitator, the guardian of the truth of the data - and the friendly face on the complexities of enterprise...

The database conversion problem outlined above could prove to be a daunting challenge if it were not for tools like those found in Omnis Studio. Its SQL Browser facility makes

quick work of this otherwise tedious and laborious process. The computer still has to do the work, of course, but the programmer doesn't have to spend a lot of time and effort on it as well. After all, isn't that why we use computers in the first place?

The "secret" is that Omnis Studio uses a specially designed facility called a Data Access Module or DAM to communicate with each database in its own syntax. This facility knows how to properly handle data of various types for each platform and translate each value between its native format and a form that Omnis Studio can use. There are also many SQL features built into objects within Omnis Studio that generate SQL code appropriate for the current database connection – even when the application is simultaneously connected to multiple and dissimilar SQL servers. Omnis Studio provides a common language, a "database Esperanto" if you will, that allows us to program the common SQL operations for an application once and then have them properly translated to the syntax for the SQL product to which an application element is currently connected.

The Omnis family of products has offered SQL database connectivity for Macintosh computers since Omnis 3 Plus version 3.3 published in 1987. This long experience with the needs of enterprise databases is rare on the Macintosh platform. With roots going back to its first incarnation in 1979, Omnis is indeed a mature technology.

Today, Omnis Studio for Mac OS X ships with DAMs for Oracle, Sybase, MySQL (commercial version only at MySQL's request), ODBC (the native DAM for Microsoft's SQL Server, but a general purpose DAM for other databases) and JDBC. FrontBase and OpenBase have also created their own DAMs to allow Omnis Studio to be front ends for them as mentioned earlier. On other platforms, Omnis Studio ships with these DAMs as well as DAMs for DB2 and Informix. There is even a special version of Omnis Studio for SAP that is ***core certified*** (meaning that applications developed for SAP using this version of Omnis Studio are ***automatically*** certified, saving the developer significant time and currency!). Should any of these database server products choose to appear on Mac OS X in the future, Omnis Studio will be ready for them.

But how does this work to our advantage as programmers and consultants? Let's perform a little experiment to illustrate the power of the DAM...

A Hands-On Experiment

Suppose that we have just been given the task of transferring data from one database product to another. We have existing data being managed in OpenBase and someone somewhere in our company has decided that we will now switch to FrontBase. (In reality, it could just as easily take place the other way around. It just so happens that a number of example databases are installed with OpenBase and not with FrontBase, so we don't have to build or install anything extra to perform our experiment...)

Of course, to perform this experiment and prove to yourself that it works as described in this article, you will need to have the right tools at your disposal. Fortunately, we have chosen software that is available for free (at least for evaluation and/or development purposes), so all you need to do is put in some time and effort to download, install and license them. And it will be worth the effort, because we will use these programs for exercises in future articles as well.

There is no room in this article to guide you through the installation of these software packages, but we can point you in the right direction. The installation process for each product is quite simple (this is all on Mac OS X, after all), so the manufacturers' instructions should see you through. Web addresses for each of these companies are given at the end of this article.

We also provide some additional setup information that you will need once you have installed the software. Again, space is a limited resource in print publications, so we must send you to the Web for this auxiliary information. There we walk you through the process of acquiring, installing and licensing these packages, starting the proper OpenBase database for the exercise in this article, creating a new empty database in FrontBase that will become a clone of the OpenBase database in our experiment and creating session templates in Omnis Studio that generically define access channels to OpenBase and FrontBase, which we will use as a basis for defining more specific channels for the exercises in this and subsequent articles. The notes you'll need are at:

http://www.davidswain.com/mactech0305swinstall.html

But whether you go off and prepare to perform this experiment for yourself or you just read through the example in a comfortable easy chair, I'm certain you'll begin to see the power and simplicity of this impressive tool!

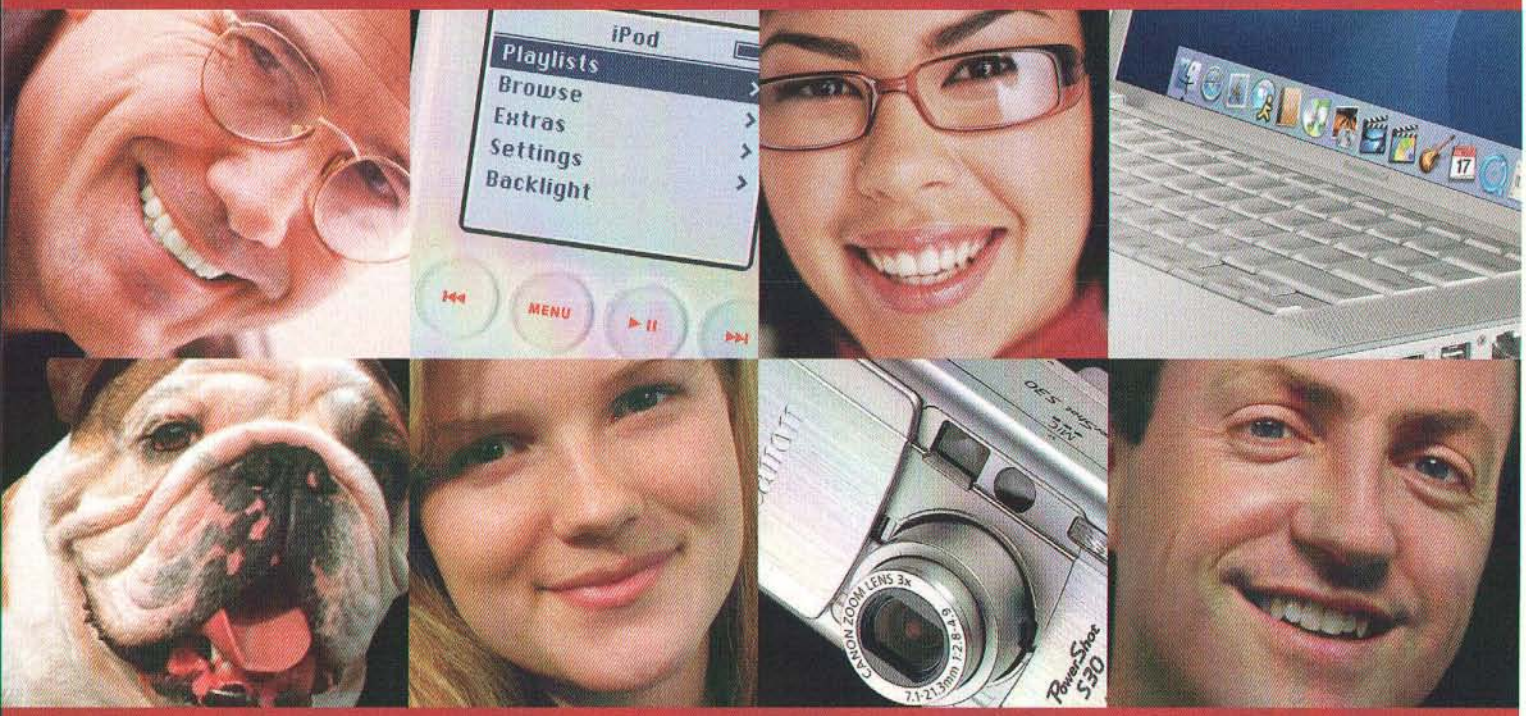
SQL Browser Basics

For this experiment, we only need to use a utility built into Omnis Studio. We don't need to do any actual programming this time, although we will execute a couple of lines of code from an SQL command line to solve some problems we will encounter in the next issue. On the other hand, it is important that you know that the Omnis Studio Integrated Development Environment (IDE) is, in fact, built using the programming features of Omnis Studio itself – so a competent Omnis programmer could build any part of this utility.

The steps and illustrations shown here are for the 3.3.3 version of Omnis Studio. Version 4.x streamlines this and related processes through the use of an integrated browser, so we don't have to open so many windows (like the Mac OS X finder) when drilling down into deeper levels of the process. If you own a copy of Omnis Studio version 4.0.2, you should have no trouble following along.

To summarize what was done in the Web exercises, we have a database named `Company` running in OpenBase. The user for this database is `admin` and there is no password. This example

At Small Dog Electronics, happy customers are our highest priority.



When you shop at Small Dog Electronics, you get more than just great selection and low prices; you also get personalized service from genuine Apple Professionals who take customer service very seriously. And that's a promise... **no if's, and's or but's.**

Small Dog Electronics

www.smalldog.com

1-800-511-MACS

Apple Specialist

A socially responsible business since 1996

database was installed with our copy of OpenBase. We created a new (and, therefore, empty) database with the same name in FrontBase, which automatically began running upon creation. The default user for that database is `_system` and again there is no password. The OpenBase and FrontBase DAMs are both nestled snugly in their appropriate locations so Omnis Studio has access to them. We are now ready to launch Omnis Studio.

After doing so, we must navigate to the SQL Browser utility. We open the SQL Browser in Omnis Studio 3.x by selecting Tools>SQL Browser... from the main menu bar. The SQL Object Browser window shown here then appears:

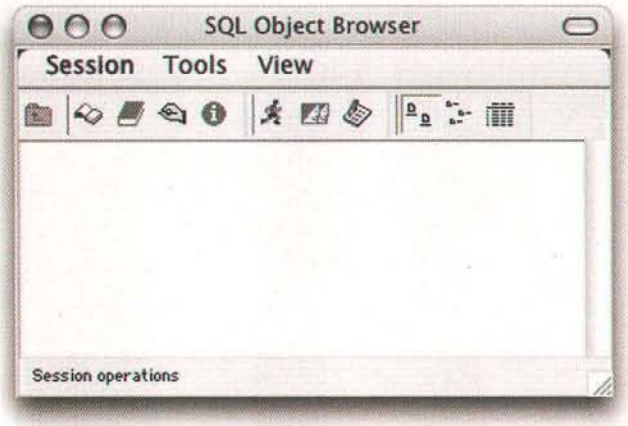


Figure 1. Omnis Studio 3.x SQL Object Browser Window

In Omnis Studio 4.x, the SQL Browser is integrated into the same window as the other browsers of the IDE and the window itself conforms to the Mac OS X standard of tools-only, no embedded menu bar. The Browser window opens when we first launch that version of Omnis Studio. But if it closes, we can open it again using View>Browser... from the main menu bar or simply by pressing Command-2. Then enter the SQL Browser by selecting the item with that name from the hierarchical tree list on the left side of the window.

The SQL Browser allows us to poke around in SQL databases for which we know valid usernames and passwords. We can also perform certain utility functions and move structural information between the table definitions in existing databases and Omnis Studio Schema, Table and Query Classes in an application library. All of these are subjects for other times. Right now we need to establish a session to each of our databases using our two DAMs so that we can manually interact with them.

In the online setup exercises, we created generic session templates for each of our SQL platforms. We will now duplicate each of these templates to create session objects that point to specific databases. OpenBase and FrontBase have their own ways of specifying login information, but we can use the appropriate syntax for each product in the fields of the Session Editor window. We get there by choosing Session>Modify Session Templates... from the menu bar embedded in the SQL Object Browser window. This opens the Session Template Manager window. The mechanics of what follows assumes the reader has performed the setup exercises, but the flow of the process should make sense whether or not the exercises have been followed to this point.

First, we'll duplicate the OpenBase generic template and open the duplicate session object. Since it is already set up to use the OpenBase DAM, we only have to specify information about the ***database*** we want to access. When logging on to an OpenBase database, we must specify the path to the database using the syntax <databasename>@<hostname>. So we will use Company@localhost here. We also know that the user whose account we want to use is `admin` and that this user does not have a password. We will also use the name `OBCompany` for this session. Our form should then look like the following figure:

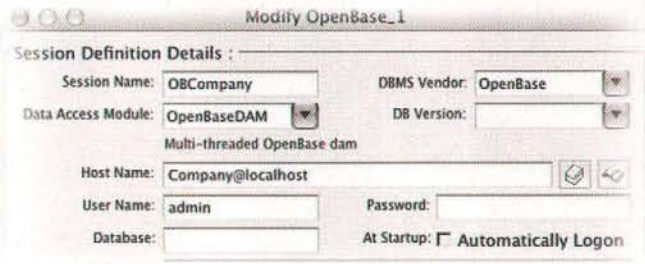


Figure 2. Session Definition for Company on OpenBase

Now we need to create a similar template to access the Company database running on FrontBase. Besides using a different name for this connection, notice that we need to use a different login syntax for FrontBase. It uses <hostname>/<databasename>, so we must enter localhost/Company. Also, we have a different user (who is still not using a password!), so we must enter this differently as well. Your template should look like this before accepting it:

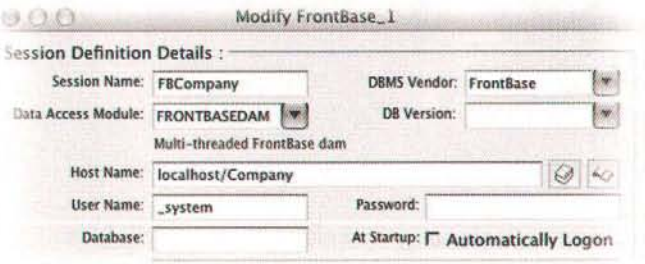


Figure 3. Session Definition for Company on FrontBase

Now let's close the Session Template Manager and return to the main SQL Browser view so we can actually connect to our databases. In Omnis Studio version 3.3.3, just close the Session Template Manager window. In Omnis Studio 4.x, just click the Back button in the navigation list.

Opening a Channel

Assuming that we specified all the correct information, opening a session to each database is a very simple process. The name of any *complete* session template will appear in the Open submenu of the Session menu of the SQL Object Browser window when it is selected. We open a session to a specific database by selecting the name we gave to that session template from this submenu.

Figure 4. Opening a Session Using a Session Template

An icon appears in this window for each open session. When both of our sessions are open, the window should look like this:

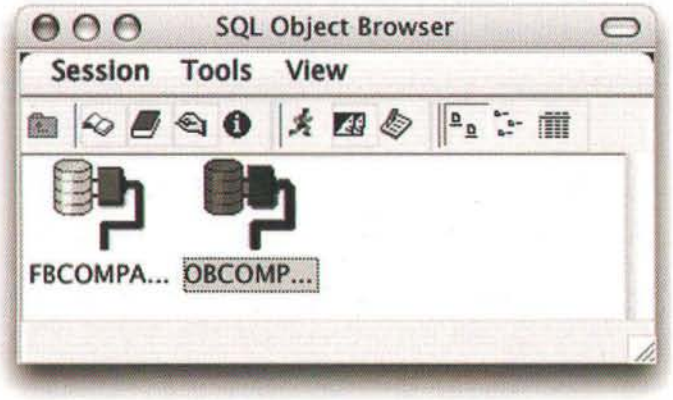


Figure 5. Open Sessions for FrontBase and OpenBase

Browsing Existing Data and Structures

We can drill down into a session to see more information by simply double-clicking on a session icon. This opens a structure category view.

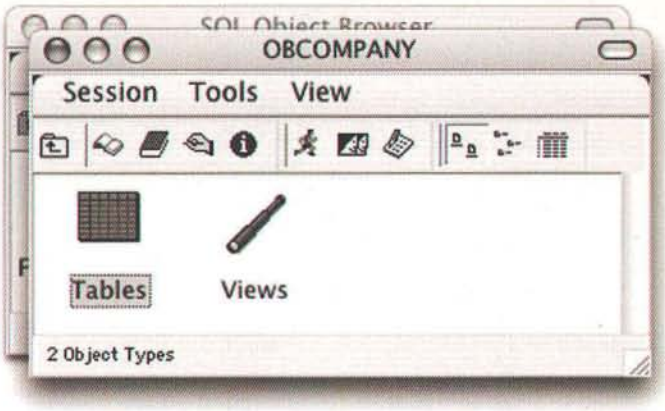


Figure 6. Structure Category View for Company on OpenBase

If we want to see the tables (and ultimately the data in those tables) in our database, we just double-click on the Tables icon in this view to expose the Tables view.

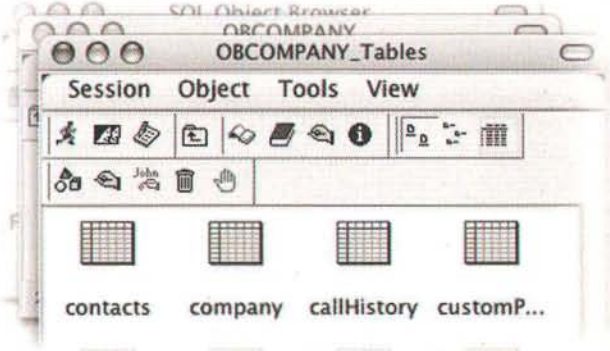


Figure 7. Tables for Company on OpenBase

Here we see that our Company database contains eleven tables. So far we haven't seen anything that can't be done in the Manager programs for either SQL product. But from here we can do a number of useful things on the Application side of the equation.

For example, we could drag one or all of these tables to our Omnis Studio application library to create equivalent table structures in the application. The DAM used to set up the connection determines which Omnis Studio data type maps properly to each column in the SQL database. These facilities come in really handy for our conversion exercise!

If we prefer to view the tables in the database as a list rather than as an array of icons, we can change to the Details view using the View menu of the window. This applies to any of the browser windows we have already encountered.

Figure 8. Switching View Modes in a Browser Window

We can view either the column structure of the data in a given table. To view this structure, we just double-click on a table icon or the line in our list view for that table.

If we instead want to view the data for that table, we can use the Show Data... item from the context menu for that table.

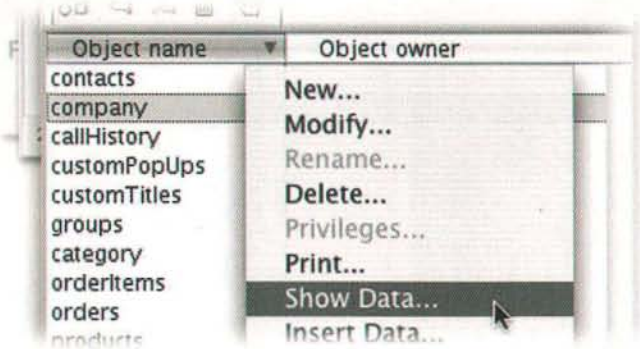


Figure 9. Show Data for company Table in Company Database

This opens the Interactive SQL window with a pre-built query to select all columns and all records for the selected table.

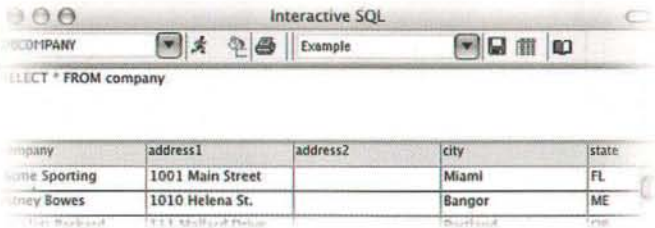


Figure 10. Interactive SQL Window with All Records for company Table

Of course, if our table contains millions of records, this may not be a good idea. But we can access this window in other ways and type in a more reasonable query for large tables. This is our SQL command line and we can use to for a number of purposes – including repairing data that won't transfer from one database to another.

OK, we've looked around the facilities a bit. Let's get down to converting that data!

Converting Data

We need to drill down to the Tables view level for both databases to perform our transfer from one to the other. Once we have done that, it's simply a matter of drag-and-drop – if all goes well. Follow along closely because we will get into a little bit of trouble once we've traversed the smooth water...

First, we will do an easy one. Place the Tables view windows for our two databases so they can both be clearly seen and drag the company table from OBCompany to FBCompany.

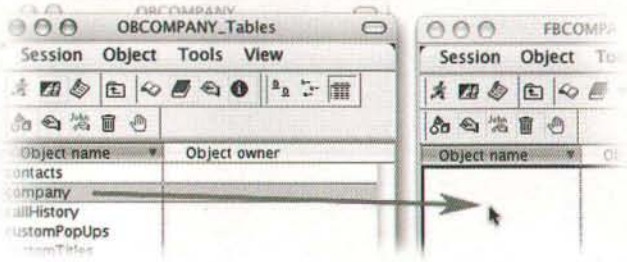


Figure 11. Drag company Table to FBCompany Session

A dialog will appear asking whether we want to transfer both the structure and the data. This is exactly what we want to do, so we just click the OK button or press the Return key to accept.

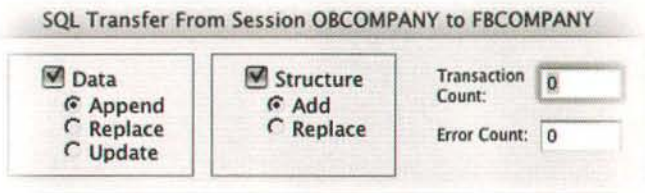


Figure 12. SQL Transfer Dialog

We are then asked to verify the column structure we wish to create in the new table for the target database. This is our opportunity to make any adjustments. We want an exact duplicate in this case, so we again accept by clicking OK.

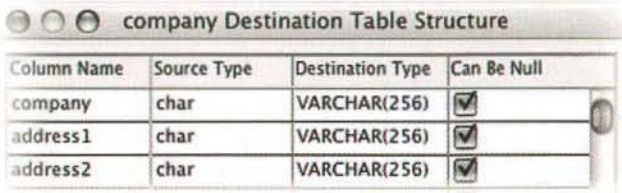


Figure 13. Destination Table Structure Verification Dialog

If there are no problems, Omnis Studio first creates a new table in the Company database and then begins to transfer all existing records. A dialog with a progress bar appears, indicating that data is now being transferred.

After the data has been transferred, another dialog appears asking us to verify the indexes for the target table. If we see no problems, we again accept.

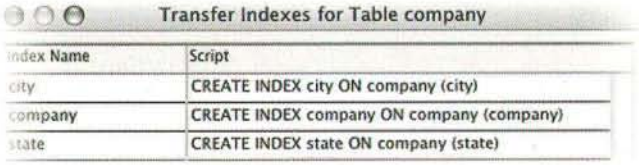


Figure 14. Index Verification Dialog

After all this activity (mostly on the part of Omnis Studio), we see our completed table now appears in the Tables view for FBCompany. Go ahead and view the data for this table to verify that it was transferred.

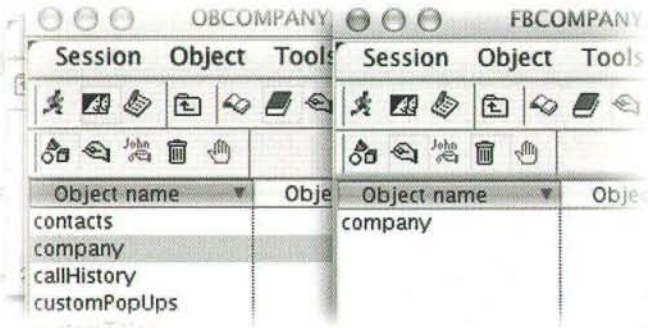


Figure 15. Completed Transfer of company Table

That was certainly easy! But we can do better than this. We can select *multiple* tables from the source database and transfer them to the target database in a single process. Follow along and give it a try. Perform these steps precisely, though, because I'm avoiding known problems that we will address and solve in the next issue...

Select all the tables in OBCompany from callHistory to products by dragging across them in the list. Then drag the whole lot of them to the FBCompany table window.

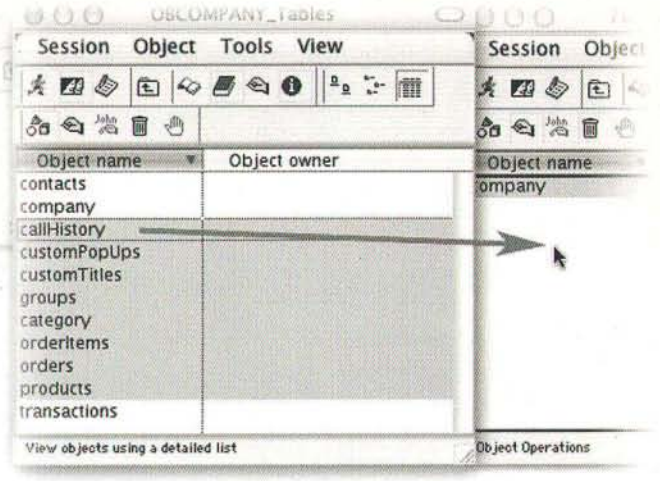


Figure 16. Drag Multiple Tables to FBCompany Session

The dialogs detailed above will open for each table in succession. Just accept each one. There should not be any problems with any of them. Notice that the user is still needed to make decisions and accept dialogs along the way, but there are few, if any, difficult decisions to make as long as the data and structure from one database is compatible with the other.

But there can be issues...

Dealing With Conversion Problems

Next time we will tackle a couple of tables with conversion problems and see how Omnis Studio makes short work of them. Try dragging the transactions table from OBCompany to FBCompany and watch what happens. It all begins pleasantly enough, just like the others did. The table structure looks fine and the data begins to transfer and then *Bang!* An error message appears part way through the data transfer! This is not a major problem, but we've run out of space to solve it this month. (Extra points for the people who solve this before they see the next article – and *double* extra points if you also solve the issues with the contacts table unaided!)

Conclusion

We have now performed a successful conversion of both table structures and data from one database to another for a number of tables. We did not have to export the data from each table into tab-delimited file, massage that data in spreadsheets to put it into the proper format for import or manually import it into the target database. We just dragged table images from one window to another. If there was a problem, the utility told us about the problem and then rolled back the transaction (structure and data transfers were separate transactions) so that we wouldn't have to deal with the additional headache of partial transfers. The utility also provides tools that allow us to solve what problems do arise – which we will explore in the next article.

Pretty cool!

So what did we learn from all of this?

First, we learned that not all SQL database products are created equally. This does not necessarily make one intrinsically better or worse than another, but it gives enterprise consumers a choice.

Second, we saw some common data conversion problems that can occur in real-life situations and we experienced potential solutions to those problems. We also had a chance to work with a useful utility in a sophisticated application development tool that gave us avenues for solving such problems.

Finally, we found out that there is a lot to learn if we want to be successful in the enterprise environment. There are new concepts and new tools to master in the process. But we learned that our favorite operating system platform is more than ready to enter the enterprise database marketplace. And it can do so proudly and effectively.

Useful Web Sites

You may wish to further explore some of the products and technologies mentioned in this article. Here are web links to sites for a number of companies who create SQL database products and related software:

FrontBase, www.frontbase.com
OpenBase, www.openbase.com
Oracle, www.oracle.com
FirstSQL, www.firstsql.com
MySQL, www.mysql.com
PostgreSql, www.postgresql.org
Omnis Studio, www.omnis.net/mactech
Actual Technologies, www.actualtechnologies.com

Bibliography and References

You may wish to brush up on your SQL skills as well. You may need them for the rest of this article series. Here are a couple of books you may find useful:

Date, C. J. and Darwen, Hugh A Guide to the SQL Standard. 4th edn. Addison-Wesley, 2000.

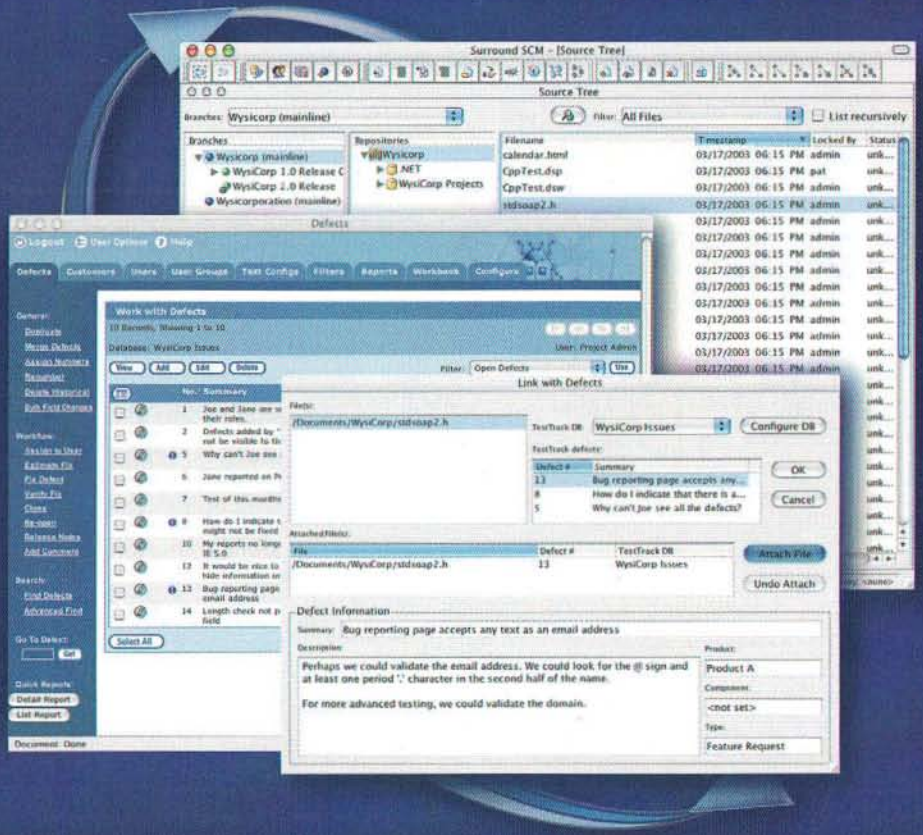
Taylor, Allen G. SQL for Dummies. 5th edn. IDG Books Worldwide, 2003.

About The Author

Since 1982, David Swain, founder of Polymath Business Systems, has leveraged his diverse background in the physical and social sciences, the business world, and the visual and performing arts to educate IT programmers and managers about the complexities of information management systems and the software used to build them. He is a regularly featured speaker at database application development conferences around the English-speaking world, which also offers him great opportunities for collecting exotic stock footage and still images for use in the video and DVD production classes he offers at his home in Bedford, New Hampshire. You can reach him at dataguru@mac.com.

Complete Source Control and Defect Management for Mac OS X

Seapine Software™
changing the world of software development



Effective source code control and defect tracking require powerful, flexible, and easy-to-use tools—Surround SCM and TestTrack Pro

- Complete source code control with private workspaces, automatic merging, role-based security, and more
- Comprehensive defect management — track bug reports and change requests, define workflow, customize fields
- Fast and secure remote access to your source files and defects — work from anywhere
- Advanced branching simplifies managing multiple versions of your products
- Link code changes with defects and change requests — know who changed what, when, and why
- Scalable and reliable cross-platform, client/server solutions support Mac OS X, Windows, Linux, and Solaris
- Exchange data using XML and ODBC, extend and automate with SOAP support
- Licenses priced to fit your budget

Seapine Software Product Lifecycle Management
Award winning, easy-to-use software development tools

Seapine Surround SCM
Seapine TestTrack PRO

software development 13th annual product excellence award JOLT

Download Surround SCM and TestTrack Pro at www.seapine.com or call 1-888-683-6456

PERFORMING BASIC IMAGE MANIPULATION...

USING YOUR EXISTING SOFTWARE!

While this concept seems amazing at first, you quickly realize that you now have thousands and thousands of image files to deal with. Those image files need to be downloaded, imported, renamed, filed, rotated, cropped, converted to other formats, and more! Sure, there are some tools that Apple gives us to help with these tasks, such as *Image Capture* and *iPhoto*. There are also a slew of other applications that you can download or purchase to aid with processing. However, wouldn't it be great if you could slap a few lines of code together to write your own application in order to help with your unique process? Well, using AppleScript, you can.

AppleScript is the perfect tool for automating many image-related tasks. As we have discussed in previous articles, AppleScript can be used to batch-rename files and folders. So, you could create a script that downloads your digital images into a custom folder structure and renames the images with the current date. You could create a script that opens up a folder of images in something like Photoshop and adds copyright information into the images' metadata. The possibilities are virtually limitless.

For this particular article, we're going to focus on performing some basic image manipulations, such as cropping, rotating, and resizing. We'll also look at ways you can convert your images to other formats, such as from JPEG to TIFF, and vice versa. Since AppleScript alone cannot manipulate images, you may be asking how will we do this? Using a background application named *Image Events*, which comes installed with *Mac OS X*, version 10.3 and higher, we can perform these tasks.

Image Events

Mac OS X, version 10.3 and higher comes with a background application called *Image Events*, which can be found in the *System* > *Library* > *CoreServices* folder on your hard drive.

> For those using Mac OS X, version 10.2, an application named Image Capture Scripting will allow you to perform a few of the same tasks that will be discussed in this article. Image Capture Scripting can be found in the System > Library > ScriptingAdditions folder on your hard drive.

The *Image Events* application can be used in conjunction with AppleScript to interact with a service in Mac OS X called SIPS, or Scriptable Image Processing Server. This service allows basic image manipulations, such as cropping and resizing to take place.

This month, we're going to talk about performing some basic image manipulations with AppleScript. With the onset of the digital camera revolution comes a new set of problems for the computer user. The first thing many people realize after purchasing a digital camera is that film is essentially free, as is developing. You can take as many pictures as you like, just as long as you have ample hard drive space to store them.

Accessing the Image Events Dictionary

In order to begin automating *Image Events* with AppleScript, the first thing you will want to do is open the *Image Events* AppleScript dictionary. This can be done by launching the *Script Editor*, and then selecting *Image Events* and clicking the *Dictionary* icon in the *Library* palette. If the *Library* palette is not visible, you can display it by selecting the *Window > Library* menu. You may also open the *Image Events* dictionary by selecting *File > Open Dictionary...* in the *Script Editor*.

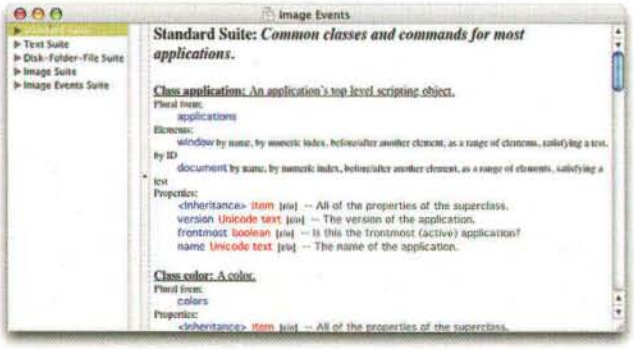


Figure 1. The Image Events AppleScript Dictionary

Once the *Image Events* dictionary has been opened, you will notice several suites of classes and commands in the left-hand pane. Some of these can be ignored. For example, you won't need to use the *Text Suite* and you may not need to use the *Disk-Folder-File Suite*. You will need to access a few commands in the *Standard Suite*, such as open and quit. You will also want to access most of the classes and commands in the *Image Suite*, which contains all of the commands for actually performing image manipulations. The *Image Events Suite* contains some *Image Events* application properties, which you may want to modify, but I will not be covering them in this article.

Opening an Image

First, when working with *Image Events*, you need to make sure that the application is running. This is done with the launch command. For example:

```
tell application "Image Events"
  launch
end tell
```

Obviously, the next step in creating a script that will manipulate an image is to open an image. This is done by passing the image path to the open command. For example, the following sample code opens a JPEG image.

```
set theImage to choose file of type "JPEG"
tell application "Image Events"
  launch
  open theImage
end tell
-> image "imageName.jpg" of application "Image
Events"
```

As you can see in the above example, the open command returns a reference to the opened image.

Accessing Properties of an Image

Once an image has been opened in *Image Events*, you may access certain properties of that image, such as the resolution, or the dimensions of the image. For

example, the following sample code gets the resolution of a JPEG image.

```
set theImage to choose file of type "JPEG"
tell application "Image Events"
  launch
  set theImageReference to open theImage
  tell theImageReference
    resolution
  end tell
end tell
-> {180.0, 180.0}
```

This is just one example of an image property that can be retrieved with AppleScript. I encourage you to explore the other AppleScript-accessible image properties, which can be found under the *image* class in the *Image Suite* of the *Image Events* dictionary. One thing to note, though, is that all of these accessible properties are currently read/only properties. They cannot be changed with AppleScript.

Performing Image Manipulations

Image Events gives you the ability to perform a variety of basic image manipulations, including cropping, flipping, padding (i.e. adding a border around), rotating, and scaling. The commands for performing these, and other, image manipulations can be found under *Image Suite* in the *Image Events* dictionary. Let's take a look at one of these commands.

The example code below shows how to scale a JPEG image to a maximum height or width, based on the image's longest side. Please take note that in order to actually see the scaled image, you must save the image after it has been manipulated. This is done by using the save command. You may also want to close the image using the close command.

```
set theImage to choose file of type "JPEG"
tell application "Image Events"
  launch
  set theImageReference to open theImage
  tell theImageReference
    scale to size 100
    save
    close
  end tell
end tell
```

One thing to note is, by using the save command in the manner above, the image you opened will be overwritten with the newly scaled image. If you want to save the newly scaled image in another location, you may optionally specify a save path.

```
set theImage to choose file of type "JPEG"
set theOutputFolder to choose folder
tell application "Image Events"
  launch
  set theImageReference to open theImage
  tell theImageReference
    scale to size 100
    save in (theOutputFolder as string) & "Scaled
Image.jpg"
    close
  end tell
end tell
```

Let's take a look at one more type of image manipulation. The following example code will flip an image horizontally, and save it into a separate folder.

```
set theImage to choose file of type "JPEG"
set theOutputFolder to choose folder
tell application "Image Events"
  launch
  set theImageReference to open theImage
  tell theImageReference
    flip with horizontal
    save in (theOutputFolder as string) & "Flipped
Image.jpg"
    close
  end tell
end tell
```

Performing Image Conversions

When working with images, it frequently becomes necessary to convert images from one format to another. Sure, you could open a JPEG image in *Preview*, and manually export it to TIFF format. But, what if you needed to do this to 100 images, or 1000? This type of process can actually be done with AppleScript and *Image Events*.

Earlier, we discussed using the save command to save images once they have been manipulated. Well, the save command also some optional parameters, one of which allows you to specify a file type in which to save the image. The following example code demonstrates how to convert a file from JPEG to TIFF format.

```
set theImage to choose file of type "JPEG"
set theOutputFolder to choose folder
tell application "Image Events"
  launch
  set theImageReference to open theImage
  tell theImageReference
    save in (theOutputFolder as string) & "Converted
Image.tiff" as TIFF
    close
  end tell
end tell
```

Image Events supports saving images in the following formats – BMP, JPEG, JPEG2, PICT, PNG, PSD, QuickTime Image. And TIFF. While *Image Events* can open GIF, MacPaint, PDF, SGI, and TGA images, it cannot save in these formats.

Pulling it Together

Now, let's pull together a few of the tasks we have discussed. The following example code uses a repeat loop to process a folder of JPEG images. Using *Image Events*, each image is opened, resized, and saved in TIFF format to an output folder.

```
set theImageFolder to choose folder with prompt
"Select a folder of JPEG images:"
set theOutputFolder to choose folder with prompt
"Select an output folder:"

tell application "Finder"
  set theImages to every file of theImageFolder
whose file type = "JPEG"
end tell
```

```
tell application "Image Events"
  launch
  repeat with a from 1 to length of theImages
    set theImage to file ((item a of theImages) as
string)
    set theImageReference to open theImage
    tell theImageReference
      set theImageName to name
      save in ((theOutputFolder as string) &
theImageName & ".tiff") as TIFF
    end tell
  end repeat
end tell
```

Other Image Manipulation Tools

As you may already know, there are some other image manipulation tools available. Both *Graphic Converter* <http://www.lemkesoft.com/> and *Adobe Photoshop* <http://www.adobe.com/products/photoshop/> offer extensive AppleScript support. By using AppleScript to automate applications such as these, you can create simple to complex automated workflows in order to perform virtually any type of image manipulation or conversion process imaginable.

In Closing

Throughout this article, we have discussed some basic ways to go about manipulating images in Mac OS X using *Image Events*. While we looked at a few examples, as I mentioned earlier, there are many other image manipulation commands and image properties that are available within the *Image Events* dictionary, that we did not have a chance to cover. The examples in this article only scratch the surface, and are meant to show you some basic ideas of what is possible. I would encourage you to begin exploring the *Image Events* dictionary in greater detail, so that you can begin using this application to its full potential, making your own workflow more efficient.

For some additional information about *Image Events*, please visit Apple's AppleScript web site http://www.apple.com/applescript/imageevents/

Until next time, keep scripting!

About The Author

Benjamin Waldie is president of Automated Workflows, LLC, a firm specializing in AppleScript and workflow automation consulting. In addition to his role as a consultant, Benjamin is an evangelist of AppleScript, and can frequently be seen presenting at Macintosh User Groups, Seybold Seminars, and MacWorld. For additional information about Benjamin, please visit http://www.automatedworkflows.com, or email Benjamin at applescriptguru@mac.com.

THE INCREDIBLES

RECOMMENDING QUICKTIME PROGRAMMING BOOKS

Introduction

One of the first questions that new QuickTime programmers ask — or *should* ask — is this: "Where do I get started?" Usually this means: "Where can I find some good documentation on programming QuickTime?" Almost without exception, the most up-to-date and most authoritative information is to be found on the QuickTime developer web pages, starting at http://developer.apple.com/quicktime. That page contains links to the latest on-line documentation, sample code, technical notes, and related materials such as QuickTime tools and information about the Apple-sponsored user- and API-level mailing lists.

The documentation links can be particularly daunting for newbies, however. There are links to over *fifty* separate documents (available in both HTML and PDF formats), which range from introductory overviews to highly-specific discussions of each major area of QuickTime programming, including sprites, movie importers and exporters, QuickTime VR, movie capture and compression, and so forth. There are also documents that assist in developing QuickTime applications in Cocoa or on Windows. So where should we begin if we want to write an image capture application in Cocoa? With the capture documentation? With the introductory documents? With the Cocoa documentation? Or suppose we wanted to write the application in Java instead; is there a good tutorial on capturing sound and video data using QuickTime for Java?

I dunno; or rather, I didn'tno until I got ahold of a copy of a new book in the O'Reilly *Developer's Notebook* series, *QuickTime for Java™: A Developer's Notebook* by Chris Adamson. In that book, Chris shows how to develop a Java application that captures video and sound data from, for example, an iSight camera. What's amazing about this is that Apple has explicitly claimed not to support sequence grabbing in QuickTime for Java versions 6.1 and later (while tantalizing us with the suggestion that "it may be provided in future releases"). Chris figured out the magic necessary to get things to work and delivers a working Java-based capture tool. Very nice. What's even nicer is that the entire book is a thoroughly readable tutorial on using Java to develop QuickTime applications.

The excellence of Chris' book set me to thinking about the general state of QuickTime programming books. If our imagined newbie wanted some recommendations on paper-and-glue books to hold in her hands and peruse from the safety of her armchair, what could we offer? In this article, I want to recommend five books on QuickTime and QuickTime applications development. Weighing in at more than 2600 pages total, I doubt that even the most persistent newbie would slog though them all before launching into a programming project. But, in my opinion, these are some of the best sources of information available to QuickTime programmers today.

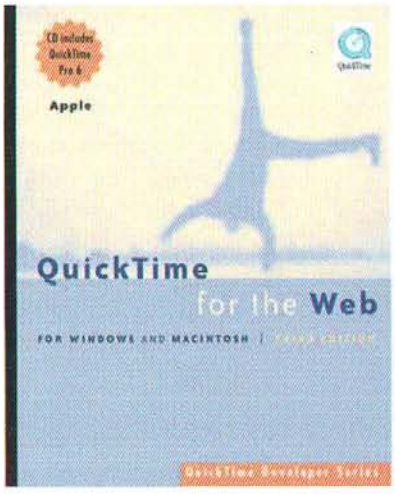
Disclaimer: I wrote two of these books, and I assisted in reviewing several others. I know all the authors personally, and I have the greatest respect for each of them. However, I have no financial stake in any of these publications. You can purchase a zillion copies of each of these books without fear that you are contributing to my retirement fund.

QuickTime for the Web

QuickTime for the Web by Steven Gulie is easily the most misnamed book in this bunch (or in just about any other bunch). From the sound of the title, you might think that it was aimed at Internet-savvy webmeisters looking to optimize delivery of their web-based QuickTime movies. Indeed, the subtitle of the first edition was "A Hands-On Guide for Webmasters, Site Designers,

One of the first questions that new QuickTime programmers ask — or should ask — is this: "Where do I get started?" Usually this means: "Where can I find some good documentation on programming QuickTime?"

and HTML Authors". Thankfully, the publisher dropped that subtitle in following editions. This book, though it does of course discuss issues involved with deploying QuickTime content on the web, is much more general and useful. It is, quite simply, the best available introduction to QuickTime as a multimedia technology. It explains in great detail the many, varied parts of the QuickTime architecture, from video and sound to QuickTime VR to sprites and wired actions. To SMIL. To Flash. To still images. To text movies. And beyond.

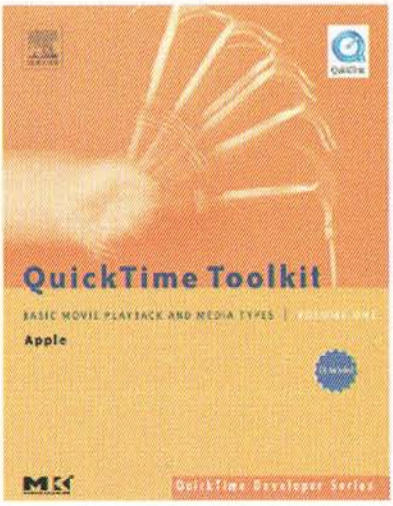


QuickTime for the Web

Now, this is not primarily a programming book. It does at one point show how to use JavaScript to control QuickTime movies playing back in a browser window, but it's not by any means an API-level investigation of QuickTime. So why am I recommending it to our newbie QuickTime programmer? Simply because a thorough understanding of the QuickTime architecture is essential for making intelligent decisions about the design and implementation of QuickTime applications. Time and again I have seen developers (and not just rank newbies) overlook easy solutions that would have been more obvious had they had a broader view of the scope of QuickTime and QuickTime movies. Steve's book provides that broad view and deserves to be the first stop on any prospective QuickTime programmer's reading list.

QuickTime Toolkit

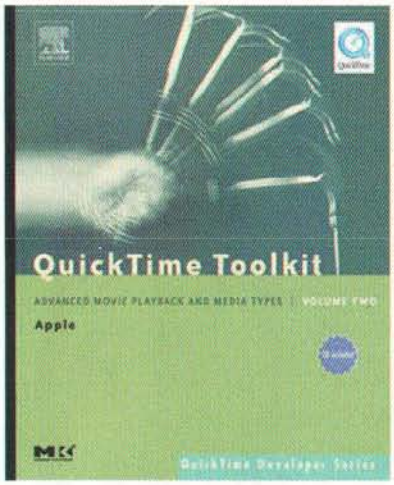
The two QuickTime Toolkit books, which I wrote, grew out of articles that appeared in MacTech magazine from 2000 to 2003. In fact, this series of articles was originally planned precisely to allow the easy transition from magazine to book form. Those books present, in my opinion, the most complete and most accessible general introduction to QuickTime programming that is available today. They also contain a fair amount of previously undocumented material that has come in handy over the years.



QuickTime Toolkit, Volume One

The first book, QuickTime Toolkit, Volume One: Basic Movie Playback and Media Types, begins by showing how to use QuickTime functions to open and display a movie in a window on the screen. It also shows how to perform basic editing operations on a movie and how to save an edited movie into a file. This book then shows how to work with a variety of media types, including video, still images, text, timecode, and sprites. It introduces concepts that are fundamental to QuickTime programming: movies, tracks, media, time scales, track references, atoms, atom containers, data references, media samples, sample descriptions, and a host of others. This first book ends with an in-depth look at

one of the cornerstones of QuickTime interactivity: wired actions and wired sprites.



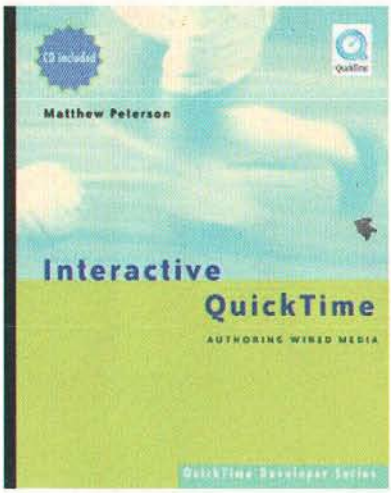
QuickTime Toolkit, Volume Two

The second book in this series, *QuickTime Toolkit, Volume Two: Advanced Movie Playback and Media Types*, continues this journey by looking at a handful of the more advanced media types supported by QuickTime: video effects, skins, Flash, and QuickTime VR. It shows how to capture movies from sound and video input sources, broadcast movies to the Internet or a LAN, play movies full screen, and load movies asynchronously. Together, these two books present a detailed narrative that covers a substantial amount of what's involved in QuickTime application programming on both Macintosh and Windows computers.

Both of these books present example code using the C programming language and reflect what we now call the "Carbon" programming model. If you are programming in REALbasic, or Revolution, or Java, or Cocoa, or any other non-C language, then you'll need to look elsewhere for introductory material geared to developing QuickTime applications in your particular development environment. However, chances are that the basic algorithms presented in these books can be transferred fairly easily to other languages. More than once, for instance, I've taken code from these books and reworked it to run in a Cocoa application. And, of course, over the past year and a half, we've had the opportunity to play with a number of these alternative development languages and tools in this column in *MacTech*.

Interactive QuickTime

Matthew Peterson is a recognized master of QuickTime's interactive capabilities — particularly of wired sprites and wired actions. I've seen audiences in awe over the amazing things he has done building intelligent, active QuickTime movies. Movies that read data off a remote server and chart graphs in real time. Movies that look and act like a pocket calculator. Movies that track the mouse and alter the appearance and location of objects in the movie based on mouse movements. Movies that simply knock your socks off, no matter how long you've worked with QuickTime.



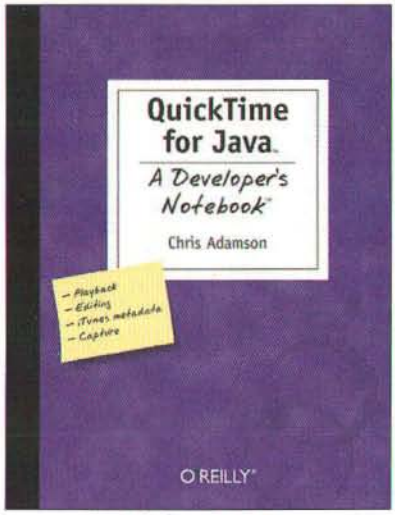
Interactive QuickTime

Interactive QuickTime is Matthew's eagerly-awaited explanation of how he does some of those amazing things. He guides us deftly through QuickTime's interactive landscape, discussing how to add dynamic, active behaviors to sprites, video, text tracks, Flash, QuickTime VR, and movie tracks. He also covers the much-neglected topic of communicating with servers using QTLists. This book is chock-full of useful algorithms and interesting approaches to programming interactive behaviors in QuickTime movies.

The examples in this book are almost exclusively presented in QScript, a scripting language supported by the LiveStage Pro application. To get the most out of this book, therefore, you need to have that tool available, and indeed the CD included with the book provides a demo version of LiveStage Pro. I can attest from long experience that building interactive movie using raw C code can be a painful process. Nonetheless, the algorithms themselves are clear enough that you could easily reimplement them in C or any other high-level language.

QuickTime for Java™: A Developer's Notebook

I know what you are thinking: "I don't do Java; I have no interest in doing Java; so this book is not for me." Fair enough. At least let me say this: if you *do* program in Java and you are interested in developing QuickTime applications, then *QuickTime for Java™: A Developer's Notebook* by Chris Adamson (O'Reilly, 2005) should be on your bookshelf. Better yet, it should be propped open on your desk as you work your way though it. We've already seen hints of the kind of careful analysis Chris brings to the table in his ability to get QuickTime for Java applications to capture video and audio sources. He shows that same insight throughout this slim volume, demonstrating how to open and display still images and QuickTime movies, support movie editing and undo operations, export movies and still images, build movies from raw data, and work with the QuickTime video effects architecture. All in Java, and all skillfully integrated with the major Java windowing toolkits, AWT and Swing.



QuickTime For Java

Actually, however, I dispute the idea that this is a book exclusively for Java developers. In several instances, Chris trots out some code that, *mutatis mutandis*, could easily be useful to C programmers. In particular, he presents a careful dissection of the format of the metadata attached to iTunes AAC files and shows how to construct a set of Java classes for reading and displaying the author, title, and album information (among other kinds). I imagine it would be easy to reimplement his algorithm in C and hence make it more widely available.

I do have one quibble with this book. This isn't your typical O'Reilly book: there's no animal on the cover and the editorial direction is much more focused. These *Developer's Notebooks* are supposed to harness "the often-frantic scribbling and notes that a true-blue alpha geek mentally makes when working with a new language, API, or project." The book design reflects this "lab manual" approach: the pages have a graph-paper ruled background and the cover shows the stains of a coffee mug. That's all well and good. But the book is printed with a bluish-purplish ink that is particularly unwelcome for the screen shots and other graphics. The text is perfectly legible, to be sure, but the images just look odd. It would have been less jarring to stick with black ink, I think.

Book Information

Books in the Morgan-Kaufmann *QuickTime Developer Series*:
QuickTime for the Web, by Steven Gulie. Morgan-Kaufmann (Third Edition, July 2003). 825 pages. ISBN: 1-55860-904-0.
QuickTime Toolkit, Volume One: Basic Movie Playback and Media Types, by Tim Monroe. Morgan-Kaufmann (First Edition, June 2004). 640 pages. ISBN: 0-12-088401-1.
QuickTime Toolkit, Volume Two: Advanced Movie Playback and Media Types, by Tim Monroe. Morgan-Kaufmann (First Edition, June 2004). 528 pages. ISBN: 0-12-088402-X.
Interactive QuickTime, by Matthew Peterson. Morgan-Kaufmann (First Edition, August 2003). 597 pages. ISBN: 1-55860-746-3.

Books in the O'Reilly *Developer's Notebooks* series:
QuickTime for Java, by Chris Adamson. O'Reilly (First Edition, January 2005). 255 pages. ISBN: 0-596-00822-8.

About The Author

Tim Monroe is a member of the QuickTime engineering team at Apple. You can contact him at monroe@mactech.com. The views expressed here are not necessarily shared by his employer.

Virtual Private Networks

Secure E-commuting

How to securely get to the office on the information super-highway

By Brad Belyeu

Introduction

Let's say your working on a big project at the office. Of course, the project is saved on the fileserver for security and accessibility reasons. The big deadline is Friday at five, and you or one of your team-members becomes deathly ill a couple days before. Or what if there was a blizzard? How in the world are you going to work on this project for your biggest client when you can't make it to the office? Create a virtual office! Using a virtual private network, you could access the network file, application, & print servers just as if you were there. When you connect to a VPN, it appears as though your computer sits right on the local network. Access is not as fast as physically being on the LAN, but when you can't be there it's the next best thing.

Defining VPN

VPNs (or Virtual Private Networks) come in many shapes and sizes. It has been a buzzword that is fairly ambiguous and has taken on different meanings over time. VPNs provide secure remote access to internal (private) networks over public networks, usually the Internet. The problem in defining VPNs has to do with its ambiguity. Virtually all networks are virtual in some sense of the word. I don't have a direct cable connection to all the Internet resources I use on a daily basis. But I get to those resources through a public network, and they are often located on someone else's private network. Besides, what one person considers private is often not private enough for someone else. VPNs can't be defined by any specific software protocols because there are several different common protocols that work effectively.

Virtual private networks can be hardware or software based. Using Mac OS X Server or a software application like VPN Tracker Server allows you to create a software VPN server, but most VPN servers are pieces of hardware external to your computer. VPN hardware is often included inside a firewall or router. For example, I use the Linksys WRV54G, which is an Internet connection-sharing router, wireless access point, VPN endpoint, & firewall all wrapped up in a single box. Cisco makes some very powerful VPN hardware for large networks. Most software servers are created for a one to one connection using client-server based technology.

There are two main types of VPNs and I will define them by the endpoints they connect. The first is the client to LAN (local area network) connection. A local area network is your private internal network. You normally use this connection for remote users to connect to a single office from anywhere with Internet access. This is

useful for telecommuting. If you have employees that work in the field but need access to office databases, this is the type of VPN that you need. The other VPN type is LAN to LAN. This VPN type is usually used to connect the resources at two office locations so they look like a single LAN. This is often used when a company has a corporate location and other locations that need access to the corporate resources, or when resources are distributed across a couple of locations.

But Why?

When companies wanted to connect different office locations together before the Internet was widespread, they normally had to lease lines from a cable company to connect the locations. This kind of a network is referred to as a trusted VPN. This was a secure way to build a network since no one outside the buildings had access to the network as long as the cable company protected their switches, but it was also a very expensive way to share resources. With the advancement of the Internet, it became clear that using public lines instead of privately leased lines could connect offices much less expensively. But could a network over public lines be trusted for private (secure) communication? Because of the need to protect data, secure VPNs were developed. Secure VPNs use network protocols to encrypt data as it leaves the originating network and then decrypt it when it arrives at its destination. If you are using a combination of both trusted and secure VPN technology, it is referred to as a hybrid VPN.

VPN Protocols

PPP (point-to-point protocol) is a protocol that allows users to dial-up a connection to access the Internet. This connection is basically a VPN. You're using public telephone lines to access the Internet, but the access is insecure. Point-to-point tunneling protocol (PPTP) is a protocol developed by several companies but usually associated with Microsoft. Microsoft added it into every OS release since Windows 98. PPTP was built on PPP technology to create a "tunnel" allowing secure passage of information. PPP is the parent technology of PPTP, and PPTP couldn't exist without PPP. PPTP sends data in encapsulated PPP packets, which are then encapsulated within IP packets. Encapsulation allows you to send different protocols over IP. That means you can send more than just IP packets over the VPN. It's like wrapping presents. You normally use different kinds of wrapping paper for different occasions, Christmas paper, birthday paper, etc. PPTP can only deliver IP packets over the Internet, which would be like demanding a present be wrapped in Christmas paper to be delivered. But if it's really a birthday present, PPTP allows you to wrap the present in birthday paper, then wrap it in Christmas paper for delivery. After the packet is delivered, the packet header knows that it is really a birthday present and takes off the Christmas wrapping paper at its final destination. That is how you can send IPX or NetBEUI packets over IP with PPTP. The trick of PPTP being able to send other packets over IP lies in the fact that it runs at the OSI (open systems interconnection) layer 2, or link layer. PPTP relies on PPP for its authentication and encryption methods. PPTP can be used to create a 'tunnel' between two locations using TCP port 1723. Once the TCP connection is established, both control messages and data packets are sent from one endpoint to the other. These control messages do everything from initializing the VPN, to keeping it alive, and closing the VPN session. Control messages also serve other maintenance functions for the VPN. The weaknesses of PPTP are that it does not provide strong encryption for protecting data nor does it support any token-based methods of authenticating users.

L2TP (layer 2 tunneling protocol) combines the best features of PPTP and L2F from Cisco Systems into one protocol. The Internet Engineering Task Force (IETF) standardized L2TP for tunneling PPP across a public network. The main two components of L2TP are the Access Controller (LAC) and the Network Server (LNS). L2TP is similar to PPTP and runs over the OSI layer 2 so it can also route other protocols through IP packets. Data is forwarded from the LAC, which can be your own computer or your ISPs, to the LNS. If you're inside a private network, L2TP may be fine by itself. Because L2TP can't prevent packets from being changed, stolen, or faked, it is usually combined with IPSec when used across the Internet.

IPSec is the most secure way to connect to a VPN network. There are two components to IPSec: the authentication header (AH) and the Encapsulating Security Payload (ESP). The AH creates a special hashing algorithm and a specific key known only to the source and destination, which is used to check for packet integrity. A security association is setup between the devices and the AH stores the output of the special computation. Then the receiving device does the same computations, it checks the AH to make sure the computations are the same. This verifies that the packet has not been altered from its original state. It is a checksum type of authentication. The AH provides authentication but not privacy; the ESP does that by encrypting the data. The ESP encryption algorithm is a key known only to the source and destination so no one in-between can decrypt the transmitted data.

IPSec can run in two different modes. These modes relate to how IPSec is set up. If two routers are creating a VPN connection, it uses tunnel mode; but if the VPN is host to host, it uses transport mode. In transport mode, only the data portion of the packet is encrypted. To obtain maximum security, one should use tunneling mode where the entire IP packet is

encrypted and authenticated. The only downfall of IPSec is that it runs on OSI layer 3 and only supports IP packets. There is a way to overcome this however. Bundling L2TP with IPSec allows a VPN with maximum security and multiple protocols. Because IPSec by itself has no way of tunneling, it is normally only used with L2TP. L2TP creates the tunnel and IPSec allows it to be secure. This is the most secure way to create a private network and is being adopted as the best standard. RFC 3193 describes the standard for bundling L2TP & IPSec. (http://www.faqs.org/rfcs/rfc3193.html)

OS 10.3 & VPNS

OS 10.3 Panther has built in VPN client support through the Internet Connect application (see figure 1). It currently allows for two different kinds of VPN connections: PPTP or L2TP over IPSec in transport mode.

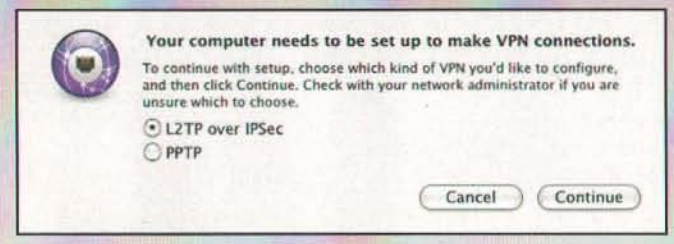


Figure 1. VPN Connections

Apple doesn't allow for plain IPSec connections because, "Pure IPSec only provides user authentication or configuration of the client machine through protocols which are either proprietary or defined by the long-expired IETF drafts, which are not standards. Using PPP/L2TP over IPSec is the only "standard" way of doing remote access with user authentication and IP address assignment over IPSec at this time." (http://docs.info.apple.com/article.html?artnum=108088) What Apple means by "Pure IPSec" is using IPSec without the internal encapsulation. The built-in functionality of the VPN client for OS X is very limited, but there are good applications for expanding that functionality. VPN Tracker (http://www.apple.com/downloads/macosx/networking_security/vpntracker.html) is an excellent application for customizing your VPN connection. VPN Tracker is one of many products developed by equinux for VPNs. VPN Tracker allows you to customize every option of the connection (see figure 2). It also has built in support for a large number of VPNs from major vendors. If you choose the option for that specific VPN server, it will automatically set all the necessary options for you.

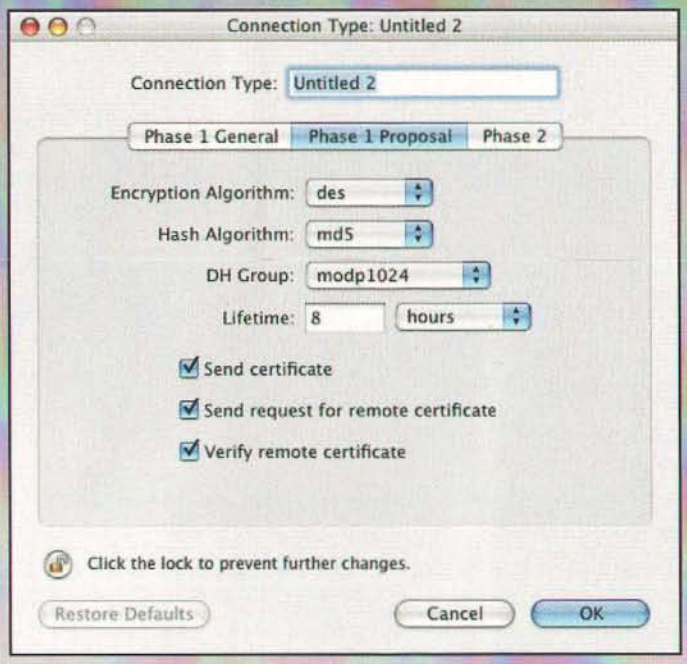


Figure 2. VPN Tracker

Equinux also makes a VPN Tracker Server in case you want to setup a VPN connection to a specific computer on your network. If you decide to setup a software VPN and you are behind a NAT (Network Address Translation) router, you need to make sure it has VPN pass-through enabled or the proper TCP/UDP ports are being forwarded.

Conclusion

Setting up a VPN can be complicated; but with the proper hardware & software, the average Macintosh user can do it. With all the different protocols, setup needs to be thought out before hand. If you choose a hardware router/VPN, research the manufacturers products first to see which protocols they work with. I've had no problem with my Linksys router, and I've setup other common brands that have worked as well (D-Link, Netgear, etc.) I would suggest viewing equinux's website at http://www.equinux.com/us/products/vpntracker/interoperability.html for a list of products that work with VPN Tracker. If you're going to use Internet Connect, it is very compatible with a large range of products; but doesn't offer near as much customization as VPN Tracker. I strongly suggest downloading and trying out equinux's VPN Tracker to keep those private networks secure!

MT

About The Author

Brad Belyeu is the President of ABConsulting based out of Oklahoma City, OK. He is an Apple Certified Technician and a member of the Apple Consultant Network.

$32 < 1 Year

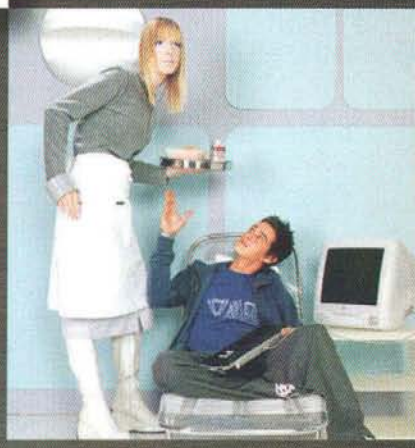
2 Years > $64

Interviews

Tapping into the world of celebrities and their Macs, only MacDirectory offers exclusive interviews. Get a close and personal view from Sarah Jessica Parker, Steve Jobs, Madonna, Harry Connick Jr., George Lucas, Jennifer Jason Leigh, Steve Woz and other leaders in the Mac community.

Features

Designers, writers, musicians, business leaders & our technical expert team offer their own personal interpretation of things that only the Mac system can deliver. With more than 200 pages of news, insights, trends and the largest Macintosh buyer's guide including over 5,000 Mac products and services.



MacDirectory

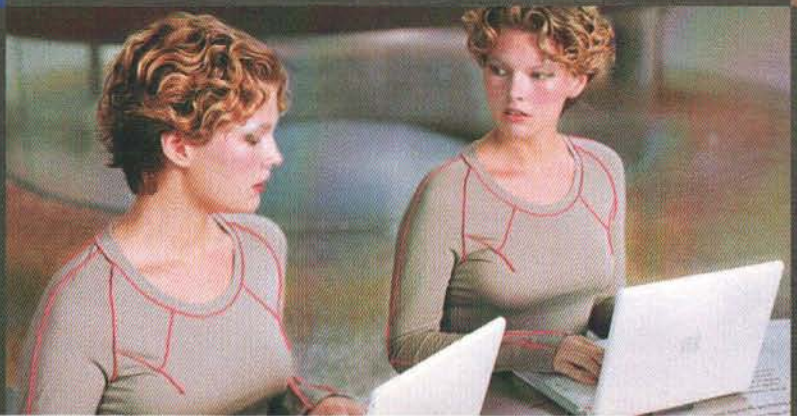
BEYOND ANY MACINTOSH MAGAZINE.

Culture

MacDirectory takes you to the wildest corners of the world and uncovers how Macintosh computers are being used by other cultures. Travel to Japan, Australia, Germany, Brazil & Russia and learn more about Apple's cultural impact around the globe.

Reviews

Find out all you need to know about the latest Mac products including the hottest Mac OS software and hardware.



Subscribe >

macdirectory.com

Send check or money order to:
MacDirectory Subscription Dept.
326 A Street, 2C
Boston, MA 02210

THE SHELL: WHAT?

IT'S WHAT YOU'RE USING IN THE TERMINAL

What in the What?

Last month, we talked about "the terminal." However, the focus was primarily on Terminal.app itself, and much less about what one actually *does* with it. Basically, Terminal.app is our interface to the shell. But what is a shell? In short, it's the user interface into Unix. It accepts input, processes said input, and is responsible for output. Much like a GUI...just with much less emphasis on the 'Graphical' component. Aside: any old NetWare users will tell you of components with the acronym 'TUI' in them. You guessed it: that's how they named their 'textual user interface' components, made to be run at the (text) console.

So a shell is really a layer of abstraction between the user and the core of the OS. Between the time of punch cards and graphical interfaces, text ruled the computing landscape. The first major shell available for Unix was the Bourne shell ("sh"), named for its creator Steven Bourne. An alternate shell, called the C shell ("csh") showed up, which took much of its syntax and semantics from the C programming language. In the Unix spirit of complete flexibility, the shell was designed to be independent from the OS. Compare that to certain OSes today that won't even let you remove a web browser without coming to a complete halt (or so we're told...).

So, what I'm getting at here is that we're going to have a lot less pictures this month. Learning to use the shell is a lot like playing an old Infocom game: you stumble around a bit, you get told that you're talking nonsense, but you ultimately map everything out and learn your way around. Even if you haven't visited every nook and cranny.

So much time, so little to do!

We're only going to cover shell basics this month, which must be understood to be built upon. You have all the time with this column as you need. The focus here will be to compare Finder actions with their command line equivalent. Next month, though, we're going to pile more on top of what is learned this month. Two things to note for this column: a) The word 'Folder', primarily used in the Finder, and 'directory', primarily used in the shell refer to the same concept: a logical container that can hold files and other Folders. I use these terms interchangeably. b) The tutorials here assume that you have a 'standard configuration' for your Mac: volumes show up on the desktop, you haven't altered your shell to start you off in some oddball location, /Users is in the default location, etc. If you've made those types of changes, I can only guess that you know what you're doing and can compensate for it here. Actually, if you made those kinds of changes, you're probably not even reading this.

As of OS 10.3, Apple has made bash the default shell. bash was created for the GNU project, and showed up in early 1988. It is a variant to the original 'sh', with a few additions. In the grand tradition of computer wit, it's only natural that there's some humor behind the name: bash stands for the "Bourne Again SHell". Just for reference, as of 10.3.7 and 10.3.8, the bash version is 2.05b.0(1).

From here on out, I'm simply going to refer to bash as "the" shell. It certainly has become very popular, and again, is now the default for Panther (and, I hope, Tiger). csh, zsh, ksh and other shells all have their purpose, and their fans. However, I'm

While there are many elements that contribute to the Zen of Unix, understanding the shell is, if not the pinnacle of this list, pretty close to it. To truly grasp not only the basics of shell use and programming, but also what is going on when you press 'return', will increase your skills considerably. This month's column will introduce us to 'the' shell, and demonstrate some substantial uses.

really happy that bash is now the default. Under earlier releases of OS X, one of the first things I did was to change my shell away from tcsh (Tenex C SHell, Apple's earlier default), and immediately change to bash. Again, that's just one of the great things about Unix: flexibility. As a multi-user OS, each user on the system can choose the shell that they prefer, and have that presented when they login.

Where it's at

bash is primarily intended for two things: interactive use, and full-blown programming (or, "scripting"). However, moving from interactive use to programming is evolutionary. Before you know it, you'll be writing scripts that automate your system and make your (or your customers') workflow easier. We're going to start with interactive commands: typing one command at a time, and getting output if appropriate. Some of this should be familiar if you followed along last month. However, this time, I give a little more detail and explain the underlying concepts in a more direct context.

Let's get to it, making one note first: as you follow along in the examples below, all capitalization and spacing is important. It makes a difference. So, if something isn't working for you, or you receive an error message where I say something will work, just double-check what you've typed.

Now, what could be more basic than opening a Finder window to peek at the contents of some directory? If you double-click on your hard drive icon in the Finder (or the equivalent of that action, like using the sidebar, etc.), you'll basically be looking at the root of your file-system. Unlike a real life tree, the 'root' of a hard drive refers to the top level: everything flows down from there. Launch Terminal.app. (You have it in your dock by now, don't you?) And there sits the cursor, blinking away. Well, just like double-clicking the hard drive icon, Terminal.app starts you out in a directory, it just doesn't display it until you ask it to. You start out in your home directory. How do you know that? Use the "pwd" command – this displays your "present working directory." In my case, typing "pwd" reveals "/Users/marczak". This shows my path from the root of the file-system. This is the same as saying, "double-click your system drive, then double-click on 'Users', and once more on the folder named 'marczak'". Now that we know where we are, let's see what's there, and compare that with what we see in the Finder.

Switch to the Finder and Apple-Shift-H to bring up your home directory. You may also want to put that window into column view, sorted by name. Back in the terminal, type "ls –l". This simply tells the bash shell to 'list' the contents of the directory you're in. The '-l' switch gives us a 'long' view; in other words, it gives some more information. This should be familiar ground if you

```
drwxrwx---  25 marczak  marczak   850      12 Jan 12:51  Applications
drwxrwx---  47 marczak  marczak   1598     8  Feb 01:01  Applications Alt
drwxrwx---  22 marczak  marczak   748      12 Jan 09:25  Audio Apps
drwxrwx---  4  marczak  marczak   136      5  Nov 07:48  Crapplications
drwx------  52 marczak  marczak   1768     8  Feb 17:05  Desktop
drwx------  56 marczak  marczak   1904     8  Feb 00:41  Documents
-rw-r--r--  1  marczak  marczak   0        14 Oct 17:54  IO.SYS
drwxrwx---  3  marczak  marczak   102      1  Feb 15:52  ISO
drwx------  44 marczak  marczak   1496     7  Feb 08:56  Library
-rw-r--r--  1  marczak  marczak   0        14 Oct 17:54  MSDOS.SYS
drwx------  23 marczak  marczak   782      3  Feb 10:20  Movies
drwx------  58 marczak  marczak   1972     1  Feb 07:17  Music
drwx------  29 marczak  marczak   986      30 Dec 07:17  Pictures
drwxr-xr-x  4  marczak  marczak   136      6  Oct 08:46  Public
drwxr-xr-x  6  marczak  marczak   204      22 Dec 17:46  Sites
-rw-r--r--  1  marczak  marczak   618      30 Dec 09:44  StCDS.zip
drwxr-xr-x  5  marczak  marczak   170      26 Oct 16:39  ZDE
-rw-r--r--  1  marczak  marczak   1044904  27 Jan 11:56  Enz Server.nfo
drwxrwx---  9  marczak  marczak   306      26 Dec 00:04  bin
drwxr-x---  5  marczak  marczak   170      31 Dec 2003   bin2
drwxr-x---  39 marczak  marczak   1326     9  Jul 2004   bin3
-rw-r--r--  1  marczak  marczak   1464     5  Feb 23:39  ca.crt
-rw-r-----  1  marczak  marczak   5715     25 Oct 08:18  edterm.term
```

followed along last month. You should get a listing like this: This file names you see in this listing should correspond with what you're seeing in the open Finder window. 'ls' can do more than this, though. Just as you may sort the Finder window by different criteria, ls can do the same. If you want to order the listing by time (or, "Date Modified"), you can add the 't' switch, like this: "ls –lt". This places the most recent dates at the top and goes back in time from there. If you want to reverse this order, you can, with the 'r' (reverse sort) switch. Try it: "ls –ltr". You should now see the contents of your home directory, sorted from oldest to newest.

If you command-click or right-click in a Finder window, you're presented with the option of making a "New Folder". bash will do this with the 'mkdir' command. While still in your home directory, type "mkdir MacTechTest" and press Enter. If you switch back to your Finder window, you'll find a new folder there called "MacTechTest". If you type "ls –l" in the terminal, you'll also see your directory listed. Now, go back to the Finder window, and move your "MacTechTest" folder to the trash. Run another "ls –l" in the terminal. You'll notice that the file is gone!

For the next comparison, please open up TextEdit (which, by the way, can be done without ever leaving Terminal.app by typing, "open /Applications/TextEdit.app"). Create a quick test file, and save it in your home directory. Call it "ShellExample.txt". Once you save it, quit TextEdit, and hop back over to Terminal. Once again, you'll see this file with "ls –l". But, you know, we should have a second copy of this file – right in the same directory. In the Finder, you'd either drag-and-drop with the option key down, or select the file and Apple-D (File->Duplicate). We can make a copy of this file in several ways, but we'll start out doing so via the 'cp' command. As you probably guessed, "cp" stands for "copy". Type "cp ShellExample.txt MacTechExample.txt". This takes the first file (ShellExample.txt) and copies it, giving it the name of the second file (MacTechExample.txt). Go check your Finder window to see both copies. One difference between the Finder and the default actions of the CLI to note here: if you copy a file, and you have a file with the same name in the destination, 'cp' will happily mow right over the destination file, replacing it with the file you're asking it to copy. Compare that with the Finder which will prompt you before doing anything like that. If you like the prompt, you can use the "i" (interactive) switch. If you try that again with the "i" switch – "cp –i ShellExample.txt MacTechExample.txt" – you'll get asked, "overwrite MacTechExample.txt? (y/n [n])". You need to type "y" and press enter for the copy to take place. Any other response cancels the copy operation, and you are notified with, "not overwritten".

Thinking about it, we don't really need both copies; let's dump one of them. Type "rm ShellExample.txt". Yep, "rm" stands for "remove". Here's an example of a command that acts differently from the Finder. When you delete something in the Finder, you move it to the trash, where it sits until you decide to "empty trash". Conversely, "rm" nukes the file on the spot. No ability to retrieve from the trash here. Once you hit that Enter key, it's gone. So, always be a little careful with the "rm" command. If you want the added comfort of a confirmation prompt, you can use the "i" switch here, too. "rm –i filename" will ask "remove filename?" Only a reply of "y" on your part will remove the file. Any other response cancels the action.

Create a test folder again: "mkdir ShellExample". Now we can move our file into the folder we just created. In the spirit of Unix economy, the command for move is "mv". Just like the cp (copy) command, you follow the mv command the names of two files: where it is, and where you want it to go. In this case, we'll move our file with "mv MacTechExample.txt ShellExample/". We can change our current directory by using the 'change directory' command: "cd". Type "cd ShellExample" and press Enter. Now, list the contents of this directory, and you'll see the file that we moved in here. Note that your *prompt* has changed to keep you abreast of your current directory. Your prompt should end with something like '~/ShellExample username$', where 'username' is your user name.

Why did I call that file 'MacTechExample.txt'? We should name it 'ShellExample.txt', like the directory it's sitting in. You rename files with the 'mv' command. Think about it: renaming is really just moving the file within the same directory and changing the destination name. Type "mv MacTechExample.txt ShellExample.txt". List the directory again and note the change.

Bring me home

We'll wrap this up with by building on the commands just covered. First, to get back to your home directory, type "cd" with no arguments. Press enter, and you're back home. Again, note that your prompt reflects this change, by just showing the tilde symbol (~). Which brings me to talk about special characters.

So far, I've shown you things that you can do both in the Finder, and via the command line. Well, what good is that? You're here to learn the power of the command line, right? Good. Let's go!

One of bash's jobs is to break apart your input into tokens, and act in some way on those tokens. Sometimes, you type something that is destined for bash itself. More often, you're really running a command that isn't built-in to bash, so bash has to fire up that command as a process, and pass it any arguments that you've supplied. Arguments are either literal – you've given a name exactly as it exists on the file-system, or, you give bash an approximation or, pattern, to match and have bash figure out the rest. For example, you've now learned that the tilde character is special to bash, and it means "my home directory." However, under the covers, bash expands this to its real value. My home is at /Users/marczak. When I type "cd ~" to go home, bash calls 'cd', but changes the parameter from '~' to '/Users/marczak', without you having to do anything. There are more characters that have special meaning to the bash shell.

Of all the characters that bash recognizes as special, the *string wildcard* character is probably the most-oft used. Since it's used elsewhere, you're most likely familiar with the asterisk ("*") being used in this capacity. "ls" is a great tool to illustrate how this works. Earlier, we simply asked 'ls' to display all contents in our current directory. However, "ls" will list specific

items when asked to do so. For instance, no matter where you are in the file-system, you can type "ls –l ~" to see the contents of your home directory. Remember again, that bash turns the tilde into "/Users/username" before 'ls' ever sees the parameter. That's a hugely important concept. "ls" itself does not have to figure out what the tilde means – it never sees it.

Make sure you're in your home directory ("cd" [enter]) and type "ls –ld Public". You'll see something like this:

```
drwxr-xr-x  4 marczak  marczak  136  6 Oct 08:46 Public
```

The "d" switch tells ls that if the argument is a directory, that you want to see the directory name, and not *its* contents. Without the 'd' switch, this is what happens with "ls –l Public"

```
drwx-wx-wx  3 marczak  marczak  102  6 Oct 08:46 Drop Box
```

See? We're shown the contents of 'Public'. However, there are situations that you don't know the exact name of the file or directory. That's where the string wildcard comes in. Try this: "ls –l P*". You should get something similar to this:

```
drwx------   29 marczak  marczak  986 30 Dec 07:17 Pictures
drwxr-xr-x   4 marczak  marczak   136  6 Oct 08:46 Public
```

I don't think you can do *that* with the Finder. Want to see which "iApps" you have installed in /Applications ? (See sidebar Figure 1.)

Conveniently, the string wildcard can be used anywhere in the pattern, and it will match any number of characters. For example, I may need to find all of my Konfabulator widget directories. I'll go to my ~/Documents directory and type: (See sidebar Figure 2.)

Notice the pattern: wildcard up front, and one in back. This way, we find the word "Widgets" no matter where it appears. One last reminder about this: it's the bash shell that takes care of dealing with the wildcard. This is called expansion. In the previous example, the bash digs through the file-system, finds the matches, and then passes that to "ls". When we typed "ls –ld Widgets", "ls" only saw this:

```
"ls -ld 'Older Widgets' 'Older Widgets December 29' 'Widgets'"
```

While I paraphrased that slightly for ease of explanation, the point is that it's the shell that deals with the special characters, and the programs and built-in commands never see that input.

One great way to test shell expansion (also known as *globbing*), is with the 'echo' command. As you can imagine, 'echo' will echo its arguments to the console. If you type "echo Hello there." and press [enter], echo will spit out "Hello there." right back at you, and then hand you back the prompt. 'echo' also respects the patterns and special characters we've discussed. To see what I mean, try this: "echo ~/P*". I get this:

```
/Users/marczak/Pictures /Users/marczak/Public
```

Naturally, you'll have your user name in place of mine. This: "echo ~/*ib*" gives me this:

```
/Users/marczak/Library
```

There are more special characters that the shell recognizes. If you'd like to see all files that begin with an e, q or v, we can use *character-set wildcards* (or, bracket wildcards). This: "ls -l /usr/bin/[eqv]*" gives us this (edited for brevity, please see sidebar Figure 3.)

```
$ ls -ld /Applications/i*
drwxrwxr-x   3 root     admin     102 15 Dec 09:25   /Applications/iCal.app
drwxrwxr-x   3 root     admin     102 12 Mar  2004   /Applications/iChat.app
drwxrwxr-x   3 root     admin     102  3 Mar  2004   /Applications/iDVD.app
drwxrwxr-x   3 root     admin     102  6 May  2004   /Applications/iMovie.app
drwxrwxr-x   3 root     admin     102 10 Aug  2004   /Applications/iPhoto.app
drwxr-xr-x   3 marczak  marczak   102 23 Jan 00:29   /Applications/iStumbler.app
drwxrwxr-x   3 root     admin     102 10 Aug  2004   /Applications/iSync.app
drwxr-xr-x   3 marczak  marczak   102 14 Apr  2004   /Applications/iTerm.app
drwxrwxr-x   3 root     admin     102 14 Jan 08:48   /Applications/iTunes.app
```

Figure 1

```
$ ls -ld *Widgets*
drwxr-xr-x   5   marczak  marczak   170    29 Dec 01:02 Older Widgets
drwxr-xr-x   17  marczak  marczak   578    10 Nov 08:54 Older Widgets December 29
drwxrwx---   30  marczak  marczak   1020   30 Dec 07:07 Widgets
```

Figure 2

```
-r-xr-xr-x   1  root  wheel    85800      26 May  2004  /usr/bin/eaytest
-rwxr-xr-x   1  root  wheel    98260      26 May  2004  /usr/bin/efax
-rwxr-xr-x   1  root  wheel    89444      26 May  2004  /usr/bin/egrep
-rwxr-xr-x   1  root  wheel    30716920   21 Jul  2004  /usr/bin/emacs
-rwxr-xr-x   1  root  wheel    37921      12 Sep  2003  /usr/bin/enc2xs
-rwxr-xr-x   1  root  wheel    292356     26 May  2004  /usr/bin/encode_keychange
-rwxr-xr-x   1  root  wheel    164756     26 May  2004  /usr/bin/enscript
-r-xr-xr-x   1  root  wheel    14044      26 May  2004  /usr/bin/env
-r-xr-xr-x   1  root  wheel    35856      26 May  2004  /usr/bin/error
-rwxr-xr-x   1  root  wheel    61164      26 May  2004  /usr/bin/escputil
-rwxr-xr-x   1  root  wheel    84784      26 May  2004  /usr/bin/etags
-r-xr-xr-x   1  root  wheel    14036      26 May  2004  /usr/bin/expand
-rwxr-xr-x   1  root  wheel    117424     15 Dec  20:33 /usr/bin/expect
-r-sr-xr-x   1  root  wheel    20072      26 May  2004  /usr/bin/quota
-rwxr-xr-x   1  root  wheel    3266       12 Sep  2003  /usr/bin/vers_string
lrwxr-xr-x   1  root  wheel    3          12 May  2004  /usr/bin/vi -> vim
lrwxr-xr-x   1  root  wheel    3          12 May  2004  /usr/bin/view -> vim
-rwxr-xr-x   1  root  wheel    967284     26 May  2004  /usr/bin/vim
lrwxr-xr-x   1  root  wheel    3          12 May  2004  /usr/bin/vimdiff -> vim
-r-xr-xr-x   1  root  wheel    15004      26 May  2004  /usr/bin/vm_stat
-r-xr-xr-x   1  root  wheel    40564      10 Feb  15:09 /usr/bin/vmmap
```

Figure 3

As expected, all of the files start with e, q or v, and the asterisk wildcard matches anything remaining. The brackets will also accept a range of characters. If you had a group of files, named by date, you could find all of the files in a directory for the year 2001-2004 with the command "ls –l 200[1-4]*". Don't forget that you can group these any way you'd like. To find dates that match 1973-1975, 1983-1985 and 1993-1995, you could use this: "ls –l 19[7-9][3-5]*".

You can also tell bash that you want everything but certain patterns. This is the job for the 'not' operator, which is represented by the exclamation point ("!"). If I want to find everything in a directory that starts with 'p', but does not end in 'sh', I'd type "ls -ld p*[!sh]", and would see:

```
-rw-r--r--  1 marczak  marczak  21251 12 Jan 11:09
printjobfail.zip
```

...even though this directory contains "printlist.sh" and "psched.sh".

Similar to the square brackets, you can use curly braces to match a list of values. The list is separated with commas. For example, if you want to list all of the graphic files in a directory, you could use "ls –l *.{gif,jpg,bmp,swf}". Note, though, that you will receive a "No such file or directory" message from ls for anything in the list that it couldn't find. echo, naturally, will just echo the expansion back at you, like this: "echo *.{sh,txt,xxx}"

```
fort.sh showds.sh upsafari.sh 1.txt mailcap.txt *.xxx
```

Note the "*.xxx" hanging on the end there, as the shell didn't find a match, it passes this to echo unaltered.

Lastly, we'll cover the question-mark wildcard. The "?" character simply matches any one character. So, in a large directory, typing "ls ?at.txt" could potentially give us:

```
bat.txt      cat.txt      rat.txt
```

There are other characters that trigger actions in bash, such as expansion, input/output redirection, job control, and more. While these characters are very helpful, they can also trip you up. When bash sees any of these characters in its input, it tries to act on them the way intended unless you tell it otherwise. One character that is particularly troublesome for people is the ampersand ("&"). For now, just know that this causes bash to run your command in the background. When there's an ampersand on the line, bash runs your command and immediately has you back the promt. Great for long running jobs that have no interaction.

But what if in the Finder you've named a folder "5&7", and you need to access it while in a terminal? Well, let's see. Open up your home directory in the Finder, and create a directory named "5&7". In the terminal, change to your home directory, and type "cd 5&7". You'll get:

```
[1] 2299
bash: cd: 5: No such file or directory
bash: 7: command not found
[1]+  Exit 1                  cd 5
```

What's going on here? Since the ampersand puts your command in the background, the first line tells you the job number that bash assigned so you can track it later on. Since the "&" character has special meaning, bash has (roughly) tokenized the input into "cd 5 &" and "7". It runs "cd 5", and tells you "No such file or directory". It tries to run a command named "7", and reports that it's not found. Finally, we're told that job number 1 has finished (exited out of its subshell).

How can you get the shell to treat special characters as normal, plain-old characters? You need to *quote* them. One way to do this is to use the backslash escape. This tells bash to ignore the special attributes of the character that immediately follows it. When you use tab-completion, you'll notice that bash backslash-escapes characters for you. Try typing "cd 5[tab]", and you'll see that bash completes with "cd 5\&7/". It escapes the ampersand by placing a backslash in front of it. It will do this with most special characters. Here's a list of characters that have a special meaning for bash:

```
*       Wildcard
?       Character wildcard
[ ]     Character-set wildcard
{ }     String-expansion wildcard / Command block
~       Home directory
`       Command substitution (backtick)
$       Variable
&       Background job
( )     Subshell
\       Quote
|       Pipe
>       Redirect output
<       Redirect input
/       Path separator
!       Logical NOT / History expansion
;       Command separator
'       Strong quote
"       Weak quote
        Space
@       Positional parameter list (only when used in
                        double quotes)
```

This is not an all-inclusive list, but it contains the characters that will cause you problems if you want them to behave without their special powers.

One interesting way to deal with this only works (currently) with Terminal.app. You can drag and drop files and folders from a Finder window onto your Terminal.app window. Terminal.app will substitute the full pathname of the object dropped, properly escaped and quoted. Pretty cool.

The other ways to quote are to use the quote characters...who'd-a-thunk, eh? The single quote is a *hard quote*. From the bash man page: "Enclosing characters in single quotes preserves the literal value of each character within the quotes. A single quote may not occur between single quotes, even when preceded by a backslash." Basically, we could replace our example above by typing "cd '5&7'".

Double-quotes are *soft-quotes*. Again, from the bash man page: "Enclosing characters in double quotes preserves the literal value of all characters within the quotes, with the exception of $, `, and \. The characters $ and ` retain their special meaning within double quotes. The backslash retains its special meaning only when followed by one of the following

characters: $, `, ", \, or <newline>. A double quote may be quoted within double quotes by preceding it with a backslash. The special parameters * and @ have special meaning when in double quotes." I'll be going deeper into this concept next month.

Getting Help

This material has been a basic, but important foundation. If you understand these concepts, you're ready for more. Plenty more is coming next in next month's column. But what about the time between now and then? You want to experiment, right? You want to see what else can be done on the command line.

Last month's column talked a little bit about tab completion. You type a few letters, hit tab, and the shell tries to complete your words with a match. Unlike the crummy tab completion in Windows, if there are multiple matches, bash beeps at you to let you know. Pressing tab again will display the matches. What we didn't cover last month is that this works with commands, too. Try this: type "ls" and press tab twice. You'll see all of the commands (in your path) that begin with "ls". Here's what I get: "ls lsbom lsof lsvfs". OK...'ls'...check, 'lsbom'....good...'lsof'...know it...'lsvf...what is that?'

There are a few ways to find out what that is. One way is the 'whatis' command. Try typing "whatis lsvfs", and you'll get this answer:

```
lsvfs(1)                - list known virtual file systems
```

Oh, OK! How does that compare to something we know? Press 'q' to get out of the listing for lsvfs and type "whatis rm". I get:

```
rm(1), unlink(1)        - remove directory entries
```

Well, yes, that sums up the rm command nicely. What does whatis have to say for itself?

```
$ whatis whatis
apropos(1), whatis(1)       - search the whatis database
getNAME(8)                  - get NAME sections from manual
source for whatis/apropos data base
makewhatis(8)               - create whatis database
```

Wow, that's a little more information than we got with the other commands. You'll notice that there are two entries that claim to 'search the whatis database'. The 'apropos' command is basically the same as whatis, except the parameter that you give it is a pattern (specifically, it's a regular expression, which, yes, will be dealt with in a future column). Unless you're careful, apropos can give you tons of output. There's two ways of getting help that are much more useful.

The bash shell itself has built in help using the 'help' command. Try it. This help is great, but only covers bash, and not the many, many, many commands that exist on the system outside of bash's built-ins. If you want to know more about lsvfs than simply that it will "list known virtual file systems", you'd use 'man'. 'man' let's you access the Unix documentation stored in the 'man'ual pages database. You tell 'man' which command you'd like to read up on: "man lsvfs" gives much better detail about this command:

```
LSVFS(1)          BSD General Commands Manual          LSVFS(1)

NAME
     lsvfs - list known virtual file systems

SYNOPSIS
     lsvfs [vfsname ...]

DESCRIPTION
     The lsvfs command lists information about the currently
     loaded virtual filesystem modules.  When vfsname arguments
     are given, lsvfs lists information about the specified VFS
     modules.  Otherwise, lsvfs lists all currently loaded
     modules.  The information is as follows:

     Filesystem  the name of the filesystem, as would be used in
                 the type parameter to mount(2) and the -t
                 option to mount(8)
     Refs        the number of references to this VFS; i.e., the
                 number of currently mounted filesystems of this
                 type
     Flags       flag bits

SEE ALSO
     mount(2), mount(8)

HISTORY
     The command from which this was derived from, as well as
     this manual, originally appeared in FreeBSD 2.0.

BSD                        January 4, 2003                   BSD
```

This is a nice, short example of a typical man page. Many other man pages can get fairly lengthy. Just as I've introduced you to the 'ls' command and some of it's switches that alter its output, please read the man page ("man ls") and understand the depth of this seemingly simple command!

Man pages exist for your reference and convenience. Take advantage of them! As form of homework for next time, read the man page for 'ditto'. It's an important Mac replacement for the 'cp' command.

In a month, then...

Like anything, the command line takes a little practice. This month's reading should ease you into usage, and point you towards ways to learn a bit on your own. Until next time, though, here's one more thing to try: without typing anything, press tab twice and answer 'y'...just look at all of the commands in there to learn about!

About The Author

Ed Marczak owns and operates Radiotope, a technology consulting company that guides clients to use technology to its fullest. In addition to the Acorn, Vax, PDP-11, Timex Sinclar and Commodore 64, Ed dabbles with the Mac.

DEV DEPOT

DevDepot has it all!

Get More out of your Mac!

DevDepot sells the tools, toys and technology to put more muscle into your Mac. Visit our online store today for special offers and great new products. **www.devdepot.com**

Television is about to get a whole lot more interesting.

NOW ONLY $324.99

EyeTV 200

The future of Television

Watch TV on your Mac!

go digital

Convert your analog video tapes to digital in realtime, then burn them to DVD! *

Watch TV on your Mac

EyeTV features a 124-channel cable-ready analog tuner and DVD quality MPEG-2 video encoder.

Record TV on your Mac and Archive to Disk

Collect and organize your favorite shows to **watch whenever you want**, then update video content to DVD.* (Toast 6 Titanium required).

Don't miss a thing

Use EyeTV's Electronic Program Guide to find exactly what you are looking for, and program EyeTV to record it. **Program EyeTV from anywhere via the Internet.**

NEW!

New Export Functionality

EyeTV now lets you export to iMovie®, iDVD® and DVD Studio Pro®, making it easier to create professional quality recordings.

from elgato

Features and Benefits

- Record TV otn Your Mac
- Edit Out Unwanted Content
- Archive Recorded TV To DVD*
- MPEG-2 Video Encoding
- Digitize Analog Video
- The Speed And Power Of **Firewire**

Programming Python

A Gentle Introduction to the Python Programming Language

By Christopher Roach

Introduction

I know what you're thinking. "With all of the languages out there why would I ever want to learn another one?" Well, if you've never used Python, and you've never had any desire to, then I'm sure this very statement can be found rolling around in your noggin at this very moment. But wait—hold on just a minute—there are several reasons for learning Python. For instance, in the essay "The Python Paradox" Paul Graham states one of the most important reasons for learning Python. In it, he states simply that "...Python programmers are smart."

Well, I'm not sure about you, but for me, the sheer fact that Paul Graham considers Python programmers to be of a higher intellectual caste is reason enough to learn the language. (In case you couldn't tell, I'm a big fan of Paul Graham's essays and his insights into the hacker community.)

Nevertheless, if you haven't found yourself totally convinced by Dr. Graham's musings that good hackers prefer Python, you may want to read on just a bit further. The rest of this article promises to go over several of the reasons why you would want to add Python to your programming toolbox.

We'll start with a short description of the Python language, and what advantages it holds for its users. Then, in the latter half of the article, we'll take a look at several code editors and IDEs and even one stand alone RAD tool, all of which will surely help make your transition from relative newbie to Python power-user an easy one.

Before we get started, however, if you already consider yourself to be a fairly erudite Python developer, don't turn away from this article just yet. As I mentioned in the previous paragraph, we'll be reviewing quite a few development tools that could easily help out even the most experienced Python developer. So, you may just want to skip the next section (since it mainly just synopsizes Python for the newbie) and proceed onto the rest of the article where you'll hopefully find some nice new toys to play with.

However, if you have no previous Python experience and you find yourself interested in learning what all the hype surrounding Python is about, go ahead and read on—the next section is just for you.

Why Python?

Let's start with a good overview of Python. Python is an interpreted, dynamic, object-oriented programming language. Sounds great right? But what exactly do all those terms mean to me? Well, let's look at the attributes mentioned above and discuss the benefits each provides.

First, Python is an interpreted language. Unlike many traditional programming languages in wide use today—such as C, C++, and Java—Python is not compiled. This has many benefits to a programmer. To start with, it means that programs can be written and executed instantly without a

costly compile step. This cuts down on the edit-test-debug cycle by allowing us to immediately see the results of changes to our code. It also means that we can write very small programs interactively (i.e., each line of code is executed as we type it to give us immediate feedback). What this means is that any small task that calls for a short throwaway program can be created on the fly (I love this feature). Plus, learning new features of the language is a snap since all you have to do is open a command prompt and run the interpreter to execute, on the fly, everything you type rather than going through the arduous process of creating a file, adding the code, compiling it, executing it, fixing it, etc.

Second, Python is a dynamically typed language. A dynamically typed language is one that does not need to know the data type of a variable at compile time, but rather it can determine the data type during the runtime of the application through the context in which it is used. Thus, when using Python, the time consuming practice of declaring all variables is avoided. If you need a variable to hold a string, simply create a variable and assign a string to it. There's no need to tell the compiler what the data type of your variable is, it can figure it out on its own. This leads to a much faster development cycle.

Finally, Python is an object-oriented language. Now, since this article assumes that the reader has at least a small knowledge of programming (and hasn't been living under a rock in the recent computing past), I'll spare you the details of the benefits found in using a language possessing object-oriented capability. I'll just say quickly that, unlike other languages, such as Perl and C++, Python was developed from the ground up with the object-oriented paradigm in mind. This means that creating a fully object-oriented program in Python is much more natural than it is in other procedural-turned-object-oriented languages.

Well, that's a nice description, but what else can Python do for me? Glad you asked. A few of "my" favorite reasons for using Python are readability, portability, and a large selection of libraries and active developers. Let's take a look at what each can do for you.

The first advantage I've named could seem to be either a curse or a godsend at first, but most users of Python have grown to think of it as the latter once they got used to it. Python has a rather quirky way of designating blocks of code that will probably drive you nuts at first if you are a C/C++ or Java programmer by trade. It designates code blocks without the use of braces or begin-end statements. Instead, Python uses indentation. This may seem like a very annoying trait at first, however, once you've had to work on a project with at least one other coder and tried to read their ill-formatted code, you'll be praising this feature in no time.

Python's use of indentation-designated blocks of code is efficient, clean, and understandable, plus it makes sure that everyone on a programming project follows an identical indentation scheme. It may take a few short programs, but once you've gotten used to it, and you use a good editor like Emacs that takes care of the indenting for you, you'll find you actually produce code a bit faster now that you are able to forget about matching up braces or typing out the words begin and end all of the time. Thus, Python ensures a certain amount of consistency across developers and provides a certain amount of brevity for the single developer.

Next, and one of my absolute favorite features of the language, is the availability of the language on nearly every platform. At home, I do all of my development on an Apple Powerbook running Mac OS X v10.3 (Panther). At work, I do all of my development on a Dell either running Windows XP or Redhat Linux. On all of these platforms I am able to use Python as my scripting language of choice. What this means is that a program I write at home can be ported over to my work computer without any fuss, and vice-versa. Throw in the availability of several cross-platform GUI libraries for Python, and you have a complete solution for developing applications that can be used on virtually every popular operating system and computer architecture in use today. This also means that programs you develop will be available to a larger audience, than say; a program developed using Objective-C and Cocoa.

The final advantage I've listed declares quite a bit in one statement. First, you should know that Python has been around for a while. It's been around long enough to have developed a strong and very loyal following (in fact, the computing powerhouse, Google, asks for Python experience in nearly all of their employees. Now that's an endorsement), but just short enough to still be considered a modern programming language.

Initially developed around 1990 at the CWI (a research institute specializing in mathematics and computer science in the Netherlands) by Guido Van Rossum, the language has been widely accepted by the programming community, especially those involved in system administration, the Open Source community, and the hacker community (as described by Paul Graham in his essay "Great Hackers "). The language has a very strong following and because of this, it is always evolving and gaining new libraries and abilities. It is definitely not a dead language, and in fact, it is currently under the control of the Python Software Foundation, a group that is dedicated to the advancement of open source technology related to the language. It's also nice to know that at the helm of this foundation is the creator of the language himself. This amount of support ensures that Python has a large community of developers available for help and questioning as well as libraries for nearly every need imaginable.

So, now that you have a good overview of the Python language and what it can do for you as a developer, you may want to take the time to actually learn the language. If you're cheap, or perhaps just broke, you're in luck. Just like any other language popular amongst hackers, Python has quite a large number of online tutorials, references, sample programs—you name it. Just about everything you need to learn the language can be found for free on the internet. You'll definitely want to check out the tutorial on the main Python website [http://www.python.org/doc/2.3.4/tut/tut.html] and after you've got the basics down, you may also want to check out the Python Cookbook [http://aspn.activestate.com/ASPN/Python/Cookbook/] to find some nice recipes for your own development.

I'll go ahead and make the assumption that since you've read this far, I've already piqued your interest enough to make you want to go out and learn the language. However, if you're going to learn Python you might as well do it in style. That's where the rest of the article comes in.

The remainder of this article is going to go over some of the code editors and IDEs that you can use on OS X to develop your Python scripts. So, get out your laptop, or fire up your desktop, grab yourself a nice cup of coffee, or whichever caffeinated life support—err...I mean beverage—you drink regularly, and get ready to start some downloading. After all, you don't want to be the last one on your block still using TextEdit to write your Python scripts, now do you?

However, Before We Get Started...

Before we get started with our journey, however, we need to make sure that we are all starting on the same page. The Mac OS, from about v10.2 (Jaguar) and above, has come with a preinstalled version of the Python programming language. However, at last check, the version that came with the operating system was v2.2. So, before we get started with our journey into the wonders of Python on the Mac, we need to download and install the latest version of the Python programming language (v2.3 as of this article's writing) which can be found at the MacPython website [http://homepages.cwi.nl/~jack/macpython/].

Once you've got the latest version of MacPython installed on your system, you're ready to start playing around with Python. But, before we can do that, there is still one last problem in need of a resolution.

Many of the IDEs that we will be reviewing in the next section of this article were developed in Python using the wxPython GUI library. So, in order for us to use these applications we're going to need a copy of wxPython installed on our system

You can download a copy of the most current library from its homepage [http://wxpython.org/], however, you'll find that if you install the most up to date edition of the library, the Boa Constructor IDE that we will discuss shortly, will not run. In order to make use of Boa, you'll need an earlier version of wxPython: wxPython v2.4.2.4. To get a copy of this version you'll need to download it from the Sourceforge repository at the following address: http://sourceforge.net/project/showfiles.php?group_id=10718.

Luckily, the install is extremely easy, since, both the current version (v2.5.2.8) and the earlier version (v2.4.2.4) have installer packages made for them. All you have to do is download the package at the aforementioned URL and follow the steps to get it installed.

Once you've got the current version of MacPython and wxPython v2.4.2.4 installed, you're ready to start experimenting with the myriad of development tools available to the Python programmer.

Code Editors And Integrated Development Environments

Well, now that we have downloaded and installed a copy of MacPython and the wxPython GUI libraries on our system, we should be able to try out some of the development environments available for Python. Several possibilities exist for the Python developer looking for an IDE or a simple code editor on the Mac and this section introduces quite a few of each.

For anyone who's looking for a simple code editor for Python, the choices are nearly limitless and most are quite good. In this category, we'll look into a couple of classic and powerful editors that already come with your system: Vim and Emacs. However, if your interests lie in finding a full IDE comparable to Apple's Xcode or Microsoft's Visual Studio, Boa Constructor, or the Wing IDE should suffice. We'll cover both of these options and we'll even look into ways that we can use Xcode as our Python development environment. Finally, we'll review a RAD tool that allows us to quickly develop wxPython-based GUIs without writing a single line of code. As a bonus, the latter application should be useful to any Python developer using the wxPython GUI library regardless of which code editor or IDE they've chosen to use.

So, lets get started by taking a look at a couple of options for those of you just wanting a simple code editor that supports Python development with simple features such as syntax highlighting and perhaps automatic indentation. The first section covers the powerful and popular editors Emacs and Vim and discusses how to get them to grok Python.

Emacs and Vim

If what you're looking for is a simple code editor, then you'll find that a few come already installed and ready for work on your computer right out of the box. Mac OS X comes with two very popular and very powerful command line editors: Vim and Emacs. Both editors understand Python and can be coerced into providing syntax highlighting with very little effort.

I fired up Vim for the first time the other day just to try it out and see if it worked with Python, and it did perfectly. In fact, all I needed to do to add syntax coloring was to type :syntax on while in command mode, and Voila, beautiful syntax coloring right from the command line. You can also add that line (sans the preceding colon) to the .vimrc file in

your home directory if you wish to always start the editor with syntax coloring turned on.

While many developers love Vim and wouldn't be caught coding with anything else, I personally find it to be a bit daunting. I learned to program on Emacs from the command line and I still prefer Emacs to just about any other code editor. It's powerful, easy to use, easy to customize, and it comes free with Mac OS X just as Vim does. Also, just like Vim, if you prefer to see your code in color, you'll need to turn on syntax highlighting by hand.

To turn on syntax highlighting in Emacs, you'll need to type M-x global-font-lock-mode once you've started the application. (Note: By M-x, I mean to use a combination of the Meta key and the x key. For the Mac, the Option key or the Escape key can act as the Meta key while in Emacs.) Otherwise, if you like to open your editor with syntax highlighting already turned on, you can add (global-font-lock-mode 1) to the .emacs file in your home directory. Make sure you include the surrounding since Emacs uses a version of the Lisp language, called elisp, for configuration and scripting.

If you've added the global-font-lock-mode line to your .emacs file and syntax coloring still doesn't seem to be working, it may be because you forgot to add the .py extension to your file and you've left out the optional she-bang line at the top of your script, thus the editor has no way of knowing that you just opened a Python file. The same is also true for Vim. You should be able to remedy the problem, however, by either adding the she-bang line to the top of your script telling the shell which program is used to run the script, or by adding the .py extension to your file's name. If you choose the former option, the she-bang line should look something like this: #!/usr/bin/python. Where, /usr/bin should be replaced with the path to the Python executable on your system, which you can find by typing in which python in the Terminal application.

However, if you have your .emacs file setup correctly and you also have the appropriate extension and/or she-bang line and yet Emacs still does not work correctly, it may be because you don't have a definition for a python-mode on your system. You may need to download an elisp file that defines the python-mode and install it. To do so, download python-mode.el from SourceForge.net at http://sourceforge.net/projects/python-mode/ and add the following lines to your .emacs file in your home directory. (Note: make sure to use the correct path to the location of your python-mode.el file in the load-file command, rather than the one I've supplied here.)

Listing 2: Adding python-mode to Emacs .emacs file

The following lines create a variable that holds the location of the python-mode.el file, loads the new mode into Emacs, and associates the .py extension with that mode.

```
(load-file "/sw/lib/python2.3/Misc/python-mode.el")
(autoload 'python-mode "python-mode" "Python mode." t)
(add-to-list 'auto-mode-alist '("\.py" . python-mode))
```

Now, I realize that not everyone enjoys using command line editors. Many programmers have grown up in a Windows environment where the idea of doing any serious work from the command line is laughable. To those of you who feel this way, I would like to suggest that you still consider giving these command line editors a try. Remember, your working with an interpreted scripting language when working with Python, and for that reason, you will find yourself wanting to use it for several small throwaway programs that will quickly run from the command line to accomplish a task or two and then be destroyed. When writing quick and disposable programs, I find it's generally fastest to develop from the command line rather than resorting to a full IDE. That said, it's not a bad idea to use an IDE when developing larger, more complex programs rather than quick scripts. For those of you who wish to use a complete IDE for those tasks, I've listed a few choices below.

Boa Constructor

Probably the most popular and most powerful Open Source IDE for Python (at least of the IDEs that work on OS X) is the Boa Constructor IDE. This program is a fully loaded IDE, with project management and a RAD tool for generating GUI code (wxPython code only). But, as good as that sounds, I did find a few problems when working with it.

First, I found it to be a little slow and unresponsive. I'm running this program on a G4, 1.25Mhz PowerBook with 1GB of RAM, and yet it still feels a bit sluggish. So, I don't see this as being a productive environment for anyone not running on a top-of-the-line Mac (preferably a G5-based PowerMac).

Second, the way the GUI was constructed disturbed me a bit. Basically, you had three main windows when starting a project, all of them disconnected from the other. So while you were working in the main editor window, the rest of the windows would gray themselves out and move to the back. This made it feel like three separate programs rather than one. There may be some way to resolve this, by turning on docking or something, but I was unable to find a way and it just left me frustrated.

Finally, Boa Constructor is written in Python using the wxPython GUI toolkit, which was our reason for download and installing the wxPython library in the last section. However, Boa Constructor does not work with the latest release of wxPython, which at this writing was v2.5.2.8. Instead, you have to download and install v2.4.2.4 of the wxPython GUI library (just as we all did during the introduction) in order to run the IDE. This is of course a minor setback and I believe that work is currently being done to update Boa to the newest release of wxPython, but for the time being you're stuck with the previous version and it is a bit annoying.

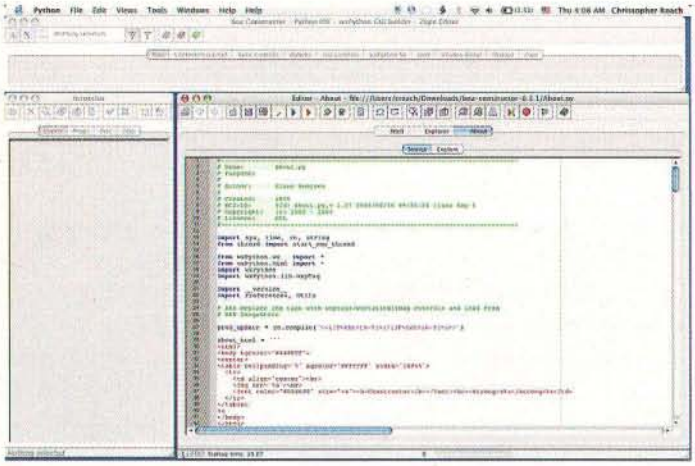


Figure 1 - The Boa Constructor IDE

In short, Boa is about the only choice you have if your looking for a full-fledged, batteries included IDE written specifically for Python, with a RAD tool and all. However, with its sluggish response and unfriendly UI, I find myself reticent to recommend it to anyone just yet. I simply feel that it's not quite ready for prime time.

Nevertheless, if you're running a top-of-the-line machine with plenty of memory and clock cycles to spare and you want a full and powerful IDE, you may as well give it a try (after all, it is free). But, for the rest of us without the means to by a dual processor PowerMac G5, I'll press on and discuss some editors that I do find worthy of looking into a bit closer.

Wing IDE

If you're looking for an IDE that's developed specifically for Python and runs like a professional tool should, then you can look no further than Wingware's Wing IDE. In the previous section I said that Boa was probably the most powerful IDE that I'll cover in this article. Well, personally I prefer the Wing IDE to Boa, and I find it to be nearly as powerful as the Boa IDE. However, one area in which the Wing IDE falls just a bit short of the Boa IDE is in its exclusion of a RAD tool for GUI development. With that said, I do believe that what the Wing IDE does...it does better than Boa, and for that reason I prefer it to Boa, or for that matter, to any of the other IDEs I've tried for Python. Plus, if you absolutely must have a RAD tool, you can always use the Wing IDE in conjunction with the wxGladeOSX GUI designer that we'll be taking a look at shortly. But, before we get to our free RAD tool, let's take a closer look at Wingware's Wing IDE.

During the time that I spent with the Wing IDE, I found it to run smoother, look better, and just feels a bit tighter than the Boa Constructor IDE. Wing IDE has code completion (one of my big requirements when using an IDE) that actually works, in other words, it helps you code rather than get in your way. Nevertheless, Wingware's IDE does have a few problems that I'll go over in the next few paragraphs.

First, the application utilizes X windows for its window management, so it will need X11 (or some other implementation of X windows) to run. Though I prefer to run my applications in Apple's native Aqua look and feel rather than using X11, the IDE still looks really great. Unlike most programs using X, it actually looks as if belongs on OS X.

Second, just like Boa Constructor, the Wing IDE was developed with Python and wxPython, and just like Boa Constructor, it runs a little slow. Surprisingly though, this one doesn't run nearly as slow as I would have expected considering the complexity of the program and of course the fact that the program is being actively interpreted rather than ran as a precompiled executable. So, while it is definitely slower than say, a compiled Objective-C program using Cocoa, it's not so slow as to make it unusable.

Third, the project management features seemed a bit clumsy when compared with other IDEs I've used. For example, when I created a new project, I was not presented with a wizard to choose the location and name of my new project, as is the case with most IDEs. Instead, I had to choose Save Project As... from the Project menu—after I had already created the project—and rename the project and choose its location. Plus, you'll need to create a folder for your new project as well, otherwise you'll find your files scattered all over the directory in which you created the project. Also, when adding a file, you're confronted with the same scenario as when creating a project. Once again, you must choose Save As...from the Project menu to designate a name and location for the file—after you've already created it—and you have to run through the entire directory structure again to place it in the same location as the project rather than the default being to place the file in the project's folder. Once you've created your file, named it, and chosen its location, it is still not yet part of the project. You must choose Add File... from the Project menu to finally add the new file to your project. None of this is especially hard to do, but it's not the most intuitive or efficient process and it's definitely not a very user-friendly design.

Finally, the Wing IDE, unlike all of the other editors and development environments I'll mention in this article, is not free. Luckily, it doesn't cost very much for a hobbyist just wanting a decent IDE on which to develop their Python applications. In fact, as of this writing, the personal edition of the IDE could be purchased for $35, and if you definitely must have an IDE, I see no reason why the Wing IDE couldn't fill that void for you.

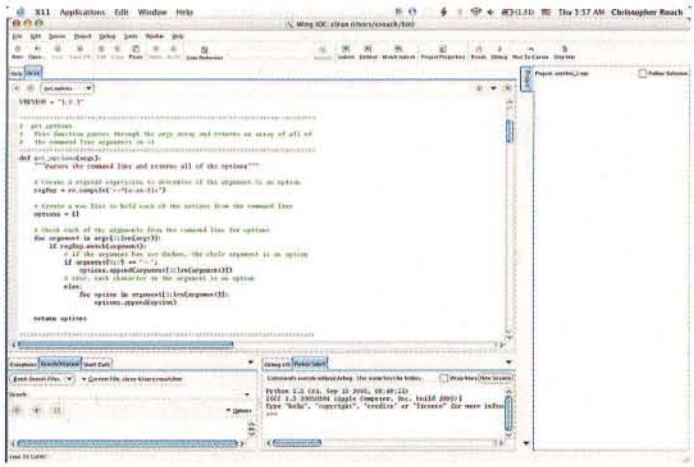


Figure 2 - The Wing IDE

All in all, this is probably my favorite of the IDEs developed specifically for the Python programming language. I wish it ran just a bit faster, and it would be nice to see someone develop a Cocoa based Python editor, but until that happens, I would recommend this IDE to anyone searching for a nice Python development tool. Plus, you can download it and try it out, fully functional, free of charge for up to ten days. So, why not take it out for a spin and make your own judgment call on whether or not it's worth the money.

Eclipse With PyDev

Eclipse is many things: a generic development environment, Java IDE, platform for tool development, GUI library—the list goes on and on. In fact, Eclipse is so large that several publishers have released entire books on just using it effectively. For our purposes though, we can just think of Eclipse as an IDE that can accommodate any language for which an extension has been developed. Python happens to be one of those languages, and PyDev is the name of the extension that provides Eclipse with the ability to grok Python.

PyDev tags itself as a complete development environment for Eclipse. It consists of three plugins. The first is the editor plugin and it supplies the user with helpful features such as syntax highlighting, smart indentation, and a whole list of the usual suspects. It claims to support code completion and refactoring as well, however, in my tests I found both of these features to be somewhat buggy. One feature of the editor plugin that I did find to be very helpful was its ability to capture syntax errors on the fly. For instance, if I tried to concatenate two strings, but I accidentally left out the '+' sign, the variable containing the second string would be underlined with a squiggly red line (reminiscent of most word processor's spell checking feature) to alert me that my syntax was incorrect and that the interpreter was expecting something other than just two operands, one right after the other.

The second plugin is the debugger. This plugin provides Eclipse with your typical code debugger, providing the user with a way to step through their code using breakpoints and look at variable values and so forth.

The third and final plugin of the PyDev development environment is the help plugin. This supplies the user with some brief, Eclipse style, documentation for PyDev.

With regards to being a proper development environment, the PyDev and Eclipse combination seems to be a formidable solution. If you're dead set on using an IDE to develop your Python applications rather than a code editor like Emacs or Vim, and yet you don't want to spend the money to purchase the Wing IDE, then I would recommend this one over any of the others I've reviewed in this article. I haven't worked with this setup for very long (as I said earlier, my favorite environment is Emacs right now) but I have used Eclipse quite extensively in the past with Java (in fact, Eclipse is my IDE of choice when doing Java development) and I have found it to be a very good IDE. Plus several major companies in the software industry have been trying to standardize on Eclipse for their IDEs. You know when companies such as IBM and Wind River (creators of the popular embedded OS VxWorks) are using Eclipse that it must be a good environment.

The main thing to remember is that PyDev is currently in version 0.8 (it has not yet reached version 1.0), and as such, it is still in its beginning stages. So, it's a little rough around the edges. Nevertheless, what it does...it does well (with the exception of features that are still in their infancy), but it does need some polish before it will feel like a fully functional Python development environment.

Xcode

Finally, I want to go over a couple of ways that you can make use of Apple's Xcode in your Python development. If you choose to go with this option you'll find that you have two schools of thought: 1) you can create pure Python applications (i.e., cross-platform, not dependent on Cocoa), or 2) you can create OS X native applications using PyObjC with Cocoa. We'll first cover creating and running a platform-independent, pure Python application, and then we'll look into what PyObjC is, and what it can do for us.

When playing around with Python one day, I finally reached a point when I thought; hey, wouldn't it be great to develop in Xcode. After all, I've done some development with Xcode, mainly just teaching myself AppleScript, but I was, nonetheless, quite impressed with the environment. It was fast and efficient and, unlike other IDEs, it didn't get in my way when I was programming. So, I started searching the web for a way to setup Xcode for Python development other than just using the PyObjC bridge to develop Cocoa based applications. After a bit of looking, I eventually stumbled across a site called pyx (Python on OS X) and it had some directions for setting up Xcode for this very task. In the next couple of paragraphs I'll share with you a quick summary of the information I garnered from that site. (By the way, the webiste's URL is http://ulaluma.com/pyx/, it's worth a look every now and then to see if any new and useful information has been reported.)

First, let's start off with developing a Python command line application. The process is pretty simple. You basically just need to setup an empty project (since Xcode does not directly support Python development) and create a custom executable to run your application. So, if you follow the steps I've detailed below, you'll be up and running in Xcode in not time.

1) Choose File->New Project to create a new Xcode project
2) Make sure you select Empty Project as the type of project to create
3) Choose where to save the project, name it, and click Finish
4) Add your main file to the application
5) Choose Project->New custom executable
6) Name your custom executable and type in the path to the python executable on your system (you can type which python in the terminal window to get this information)
7) Double click on the custom executable you created to open its main information window.

8) Click the arguments tab and click the '+' button to add a new argument to the list.
9) The new argument should be the complete path to the main file of your application (you can either find this path using Finder or the Terminal application)
10) Choose Debug->Run Executable or Press opt-command-r to run the application.

Running a Python application that uses the Window Manager is essentially the same as running one from the command line, however, we need to tell Xcode to use pythonw instead of python (pythonw is needed to run a Python script that makes use of the Window Manager, i.e., a GUI-based application). This is where we run into a small problem. It seems that Xcode only wants to run a binary executable, which python is, and pythonw most definitely is not. Since pythonw is a simple shell script we can run it by passing it and the file we wish to execute as arguments to the sh program. Using this method, our new executable path should look something like the following: /usr/bin/sh /usr/bin/pythonw /path_to_main_file/filename.py.

Another way that we can run our GUI-based application is to put the full path to the executable that is referenced within the pythonw script into the executable path of our custom executable that we created in our Xcode project. If you prefer this method, you'll first need to get the location of the pythonw script, which you can do by typing which pythonw into a Terminal window. Once you've got the location of the pythonw script, you'll need to open the script with your favorite editor or display the contents of the script with the cat program and copy and paste the full path to the Python executable from the pythonw script into the executable path of the custom executable we created to run our application. Once you've done this, you'll be able to run your GUI-based Python programs directly from Xcode.

If you like Python and you don't really care to create platform agnostic applications, you may find that developing Python applications with PyObjC is the most logical environment for you. PyObjC is an Open Source project that provides a bidirectional bridge between Python and Objective-C. Thus, Python programmers have a way of accessing the classes of Apple's Cocoa Framework through this bridge. One of the most important uses of this is to provide the Python programmer with access to the Cocoa GUI classes allowing him/her to write OS X native, GUI-based applications in pure Python.

My experience with PyObjC has been promising, though I still believe that it needs a few more iterations before reaching the usability I prefer in a development environment. For example, I wish it were possible to generate the python files from Interface Builder as you can for Java and Objective-C rather than going to the command line to do so. Small gripes such as this aside, I did enjoy developing applications using PyObjC. I found it to be very usable, very stable, and very easy to install. In fact, binary installers are available for both Jaguar and Panther editions (v10.2 and v.10.3) of OS X and can be downloaded from the projects main website [http://pyobjc.sourceforge.net/].

If you're planning on developing applications for other platforms as well as OS X, then obviously this is not the option you should go with. However, if you enjoy programming in Python and you're looking for a way to use it to develop OS X native applications, or if you're just looking for an alternative to Objective-C, then PyObjC may just be perfect you.

wxGlade

Before we close out our discussion of code editors and IDEs, I wanted to go over one final option that is neither code editor nor IDE, but it does seem to fall into a similar category. The wxGlade designer is a RAD tool that can be used to design GUIs for your Python applications using the wxPython library.

The application was created using Python and the wxPython GUI library. It can be used to rapidly construct a graphical interface using the drag-n-drop methodology that other popular RAD tools use. Once the GUI is completed, wxGlade will output the code used to create the GUI. This code can then be plugged into your application and you can then go about adding the code that does the actual work saving you countless hours writing the code that creates the applications GUI.

The designer isn't just limited to producing code for Python, however. The wxGlade GUI designer also produces wxWidgets code for the C++ and XRC (an XML based code format) languages as well. This means that, should you ever become interested in using your knowledge of wxWidgets to develop cross platform applications based on another language such as C++, (say, if more speed is needed) then you'd already have the ability to do so. So, basically more bang for the buck—nice, eh?

The wxGlade GUI designer was based upon Glade, the GTK/Gnome GUI Builder. It's definitely not the most intuitive GUI development tool to use, so it will take a bit of training to become an expert with it, but it's free, and once learned, it does speed up the process of writing a GUI quite a bit. Also, it has a very simple install since a binary (called wxGladeOSX) has been created and can be found at the WordTech website [http://www.wordtech-software.com/wxglade.html].

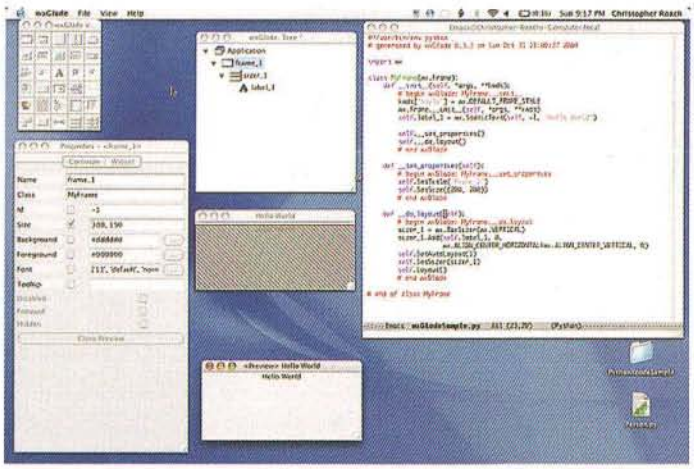


Figure 3 - wxGladOSX With Code Displayed in XEmacs

Conclusion

As the title would suggest, this article was meant to be a very gentle introduction to the Python language and to what Python has to offer Macintosh developers. Hopefully, if I've succeeded in my job, you'll come away from this article with an interest in learning Python and you'll be armed with a whole slew of development environments to aid you in the process. If this is the case, make sure that you tune in for my second article, which promises to probe deeper into the tools available to the Python developer on OS X.

Unlike this month's article which was geared towards Python newbie's, next month's article will be geared to those of you already employing Python in your development routine. Those of you who fall into this group, will be introduced to several libraries that will make your use of Python easier and more effective.

So, until then my new Pythoneers—best wishes in your Python development, and happy scripting.

Bibliography and References

Graham, Paul "Great Hackers" May 2003, http://www.paulgraham.com/pypar.html (15 Nov. 2004).

Graham, Paul "The Python Paradox" Aug. 2004 http://www.paulgraham.com/pypar.html (15 Nov 2004).

http://www.python.org/

http://aspn.activestate.com/ASPN/Python/Cookbook/

http://homepages.cwi.nl/~jack/macpython/

http://ulaluma.com/pyx/

http://boa-constructor.sourceforge.net/

http://www.eclipse.org/

http://pydev.sourceforge.net/

About The Author

Christopher Roach recently earned his MS in Computer Science from Virginia Tech and currently works as a software engineer in Florida's Space Coast. On the weekend he tries to find time to write articles on Macintosh programming and do battle with insanely powerful hurricanes, while still trying to preserve some semblance of a life. If you have questions or comments on the article, you can email him at croach@vt.edu.

Website Not Found By Clients
HTTP 404 - Website not found

The website you are looking for can't be found by your clients. It may have been improperly marketed, had poor design, or didn't work. Regardless, it's not helping your business!

Your Options:

- Rent a chicken suit and stand on the corner handing out flyers
- Paint the company URL on your chest and face during a major sporting event
- Contact SharpNET Solutions internet marketing and web design specialists, watch your traffic and rankings increase, get great feedback from all your new customers.

HTTP 404 - Website needs SharpNET

A Smart World After All

At the time, I wasn't what one would call a "computer professional," though I did some consulting work in the area of Desktop Publishing, training on QuarkXpress and Illustrator, and managed to subsidize my desire to become a poet while suffering as a poor grad student. Though I was quite aware of certain signs of hard disk trouble, such as "sticktion," which would delay startup, or the "whine," which indicated bearings that were about to go, my trusty PowerBook 520c exhibited no signs of any disk problems, not one disk error, not one bad sector. Its hard drive died suddenly, without even a hoarse whisper. I'm really not waxing poetic; back in the mid 90s, the only real clues that a hard drive was dying (other than the occasional -36 disk error or bad sector turned up by a disk utility), were audible (a scratch, a ping, a scrape), so the hyperactive herky-jerky animation of Dr. Peter Norton rubbing his stethoscope over a hard disk platter in Norton Utilities 3 wasn't really that far from the reality of troubleshooting a failing disk.

It was rather amazing how accepting I was of the situation. I turned in my books and moved on to a career in Macintosh technology. I took responsibility for being a dummy and not backing up. I never once thought to ask, how come my Mac didn't alert me that there was a problem with my disk? In 1996, we didn't have such high expectations of our personal computers. We wanted them to work, to print, and sometimes to connect to a network. A cell phone or pager was a luxury, and email accounts were generally bundled with a job, or student enrollment at a University, or an AOL or CompuServe membership.

Besides my life-changing data loss, it seems that 1996 was a pivotal year for hard disks in general. In April of that year, a group of researchers from several drive manufacturers was busy hammering out version 2.0 of SFF-8035i, a standard for hard drive diagnostics which had been proposed the previous year, defining thirty attributes related to performance and reliability that hard disks should track internally. The standards they developed became known as the Self-Monitoring, Analysis and Reporting Technology systems, now referred to as SMART.

Disks Will Die

One of the biggest issues Veterinarians have is that the pets they treat don't have the ability to communicate specific symptoms of disease. Dogs and cats with organ failure generally don't complain about pain, but simply seem to slow down, not run as fast, or jump as high. No matter how perceptive the Vet, or how experienced, I've often heard them exclaim, "I wish the animals could tell me what was wrong." How strange would it be if, like an automobile, the pet had a diagnostic port that would reveal a score about how well their

Having a hard drive die on you can be a life-altering experience. I'll never forget the first time it happened to me. It was 1996, and I was sitting, drinking an espresso in a local coffee house, working on an essay for my Ph.D. comprehensive examinations in English, when my PowerBook 520c froze. I didn't think much of it at the time, as I was running System 7.5, which tended to freeze without warning, but when I rebooted, I found myself staring dumbly at the blinking question mark. Hours later, after running Norton Utilities, and seeing that my 160 meg hard drive was now recognized as an 80 meg hard drive, I realized that my essays and research were lost. Sure I had a two-week old backup, but I'd been in a frenzy, working hard to make a deadline, and hadn't thought to keep my backup current. I took it as a sign, and even though I'm occasionally called "Dr. Dean" when I show up on-site to troubleshoot an ailing Mac, there's now no other reason to call me Doctor.

heart or kidneys, or intestines were functioning? Automobiles have such diagnostic ports, and some older (much much older) Macs used to have them as well.

Hard disks are mechanical devices, just like Zip disks, or floppy disks. They have motors. They have platters that rotate at speeds that would make the engines in most cars overheat and explode. They have little arms with magnetic heads that dance around picking up blocks of data. Frankly, it's an amazing testament to the engineering skills of hard disk manufacturers that they are as reliable as they are. With OS X, hard disks will get even more of a workout if there's not enough RAM installed in a machines, due to virtual memory page outs, leading to a premature demise of the startup disk.

Most end-users and far too many system administrators don't realize that current hard disks are keeping an internal log of their own performance, based on the SMART attributes codified and updated with the ATA-3, 4. and 5 standards. Modern SCSI disks also have SMART capabilities as well. Some of the attributes that SMART tracks are: (see table)

SMART attribute implementations still vary, somewhat, by manufacturer, but they have enough in common to, at the very least, give a pass/about to fail status report when queried. Starting with OS X 10.3, Apple's own Disk Utility includes a simple SMART test, which will display a one-line S.M.A.R.T. status report when you select a physical disk:

ID#	*Attribute*	*Description*
1	Raw Read Error Rate	Count of non-corrected read errors. More errors (a smaller value) denotes a deteriorating disk surface.
2	Throughput Performance	Throughput (I/O) performance of Hard Disk.
3	Spin Up Time	Spindle spin up average time (from parked to ready).
4	Start/Stop Count	Cycle count of spindle start and top.
5	Reallocated Sectors Count	Count of reallocated sectors. A great indication of a failing disk. Reallocated sectors are marked as "unusable," which is why modern hard disks don't show bad sectors.
6	Seek Error Rate	Count of seek errors. This indicates errors when the heads have a mechanical failure or when the heads positioned over a data block cannot read it due to poor disk surface conditions.
7	Seek Time Performance	If this attribute is lower, it indicates mechanical problems with the heads or disk surface.
8	Power-On Hours	Total of many hours the drive has been powered up.
9	Internal Temperature	How hot the temperature sensors say the drive is. Can also be a great indicator of how hot the inside or a

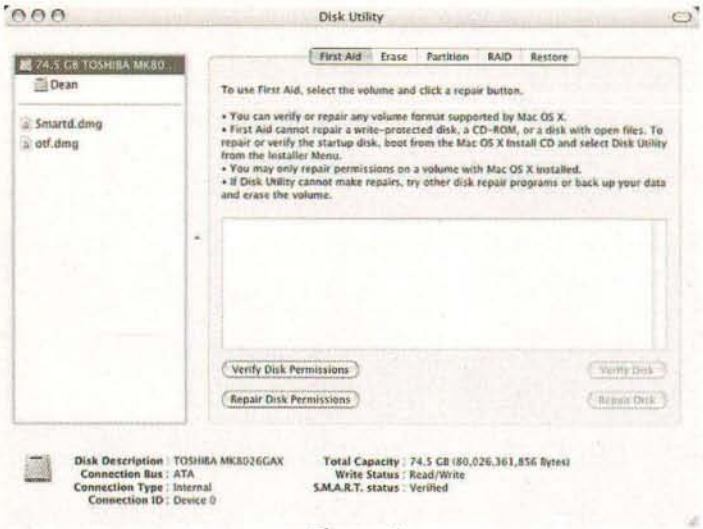


Figure 1.

The SMART test is also available from the Terminal in OS X, via the diskutil command, which is specific to OS X. First, you have to get the hard drive identifier:

```
minime:~ dean$ diskutil list
/dev/disk0
   #:  type name              size        identifier
   0:  Apple_partition_scheme *74.5 GB    disk0
   1:  Apple_partition_map    31.5 KB     disk0s1
   2:  Apple_HFS Dean         74.4 GB     disk0s3
minime:~ dean$
```

In this case, the hard disk identifier is disk0, where as the volume identifier is disk0s3. Working with the hard disk identifier, we can then query diskutil to get info on the drive, which will include the SMART Status:

```
minime:~ dean$ diskutil info disk0|grep -i smart
   SMART Status:        Verified
```

Note that in piping to the grep command, I use the –i switch to turn off case sensitivity. Diskutil is very useful for shell scripts, such as the one I wrote below:

```
#!/bin/sh
## This script is designed to get the SMART status of a drive and send an email notice on fail
## to test, change the "good=1" below to "good=0" and you should receive a warning email
## Dean Shavit, MOST Training & Consulting dean@macworkshops.com http://www.macworkshops.com
## This is for OS X Machines with ATA drives only
##
## Step 1: Define a variable for a functional drive by counting the number of SMART Verified Disks
status=`diskutil info disk0|grep -ci verified`
## Step 2: Define a number for comparison against a failed drive
good=1
## Step 3: Define the warning message for the body of the email
warning="houston we have a problem!"
## Step 4: Define a variable for the computer name
box=`/usr/sbin/scutil --get ComputerName`
## Step 5: Define a variable - email address of person to notify
admin="dean@macworkshops.com"
## Step 6: compare current status with good status, if a match, echo, if not, notify
if [ $status == $good ]; then
echo $status
else echo From: $box- $warning!!! > /tmp/houston.txt| mail -s "SMART Alert Report" $admin < /tmp/houston.txt
fi
done
```

This script, when run as a cron job, will check the hard disk periodically for SMART status and email you when the drive when the drive has a problem. There are two status codes that Disk Utility can report: "Verified" if none of the SMART attributes exceed their normal thresholds, or "About to Fail" which will appear in red letters, indicating that the drive will fail and should be replaced. Of course, "failed" isn't a status that the drive can report, but is a state! It is important to understand that even if the hard disk passes the SMART test, there's always the chance that the drive might suddenly expire, without ever issuing a warning. The SMART tests are diagnostic tests, and should never be used as a replacement for a good backup strategy. If anything, the advance warnings will help indicate when a drive needs to be replaced, so as to minimize any possible downtime of a server or workstation.

For those who don't want to script or want an easier path to SMART notification, there's a great donationware utility called SMARTReporter downloadable at http://homepage.mac.com/julianmayer that provides a nice Cocoa GUI for accomplishing the same scheduled diagnostic and warning email, without having to use the Terminal. For those consultants or admins who support off-site users, this can provide an invaluable early warning of a drive failure.

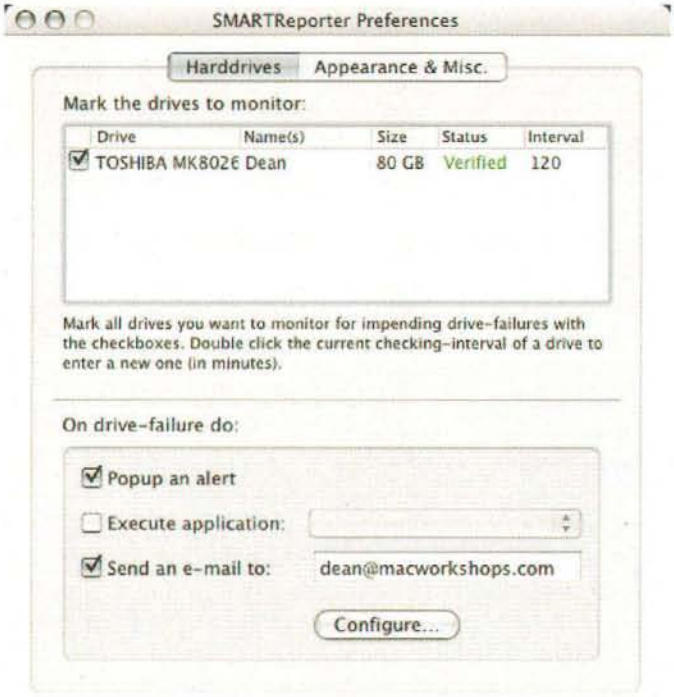
SMARTReporter Icon

SMARTReporter's a snap to install, just drag and drop it from the its disk image (.dmg) into your Applications or Utilities folder. When running, it can provide a SMART status indicator on your menu bar:

SMARTReporter Status Indicator

SMARTReporter's preferences allow an admin to change the interval of the check, define an icon set for the menu bar, set up email information, even launch another application if there's a hard disk failure, such as special Applescript Applet to throw up an alert to the user or give instructions on what to do. For email alerts, it can either use

the mail information specified, or borrow the Apple Mail.app settings. If I had my druthers, Apple would include much the same functionality in future versions of Disk Utility.



SMARTReporter Preferences

Not So Smart

SMART Reporting in OS X does have its limits, however. SCSI and FireWire disks aren't supported, so if an admin is predisposed (like I am) use SCSI drives for internal RAID mirrors on servers other than Xserves, the diskutil command cannot get the SMART status of an SCSI disk:

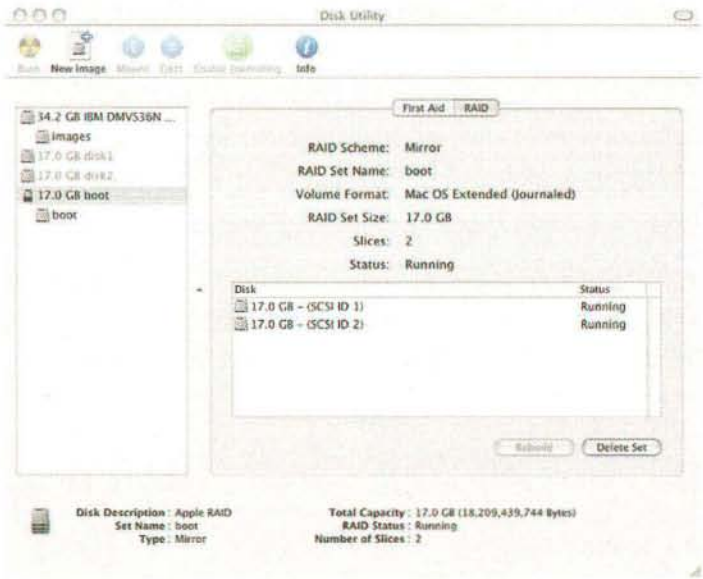
```
host2:~ mostadmin$ diskutil list
/dev/disk1
   #:  type name                    size       identifier
   0: Apple_partition_scheme        *17.0 GB   disk1
   1:    Apple_partition_map        31.5 KB    disk1s1
   2: Apple_Driver_OpenFirmware     512.0 KB   disk1s2
   3:         Apple_Boot_RAID       17.0 GB    disk1s3

host2:~ mostadmin$ sudo diskutil info disk1
   Device Node:         /dev/disk1
   Device Identifier:   disk1
   Mount Point:
   Volume Name:

   Partition Type:      Apple_partition_scheme
   Bootable:            Not bootable
   Media Type:          Generic
   Protocol:            SCSI

   Total Size:          17.0 GB
   Free Space:          0.0 B

   Read Only:           No
   Ejectable:           No
   OS 9 Drivers:        Yes
   Low Level Format:    Not Supported
```

Note that the limitation for SMART reporting to ATA disks is something inherent in the way that OS X treats disks. FireWire disks, though, which always contain ATA or SATA drives running on Firewire bridge boards, don't report SMART status to diskutil either. This can be a major problem for Mac admins, who are increasingly relying on storage devices for second-tier file services or in some cases, low-cost backup devices. In the arena, all enclosures are definitely not created equal, as shown by Granite Digital's Firevue™ drive enclosures (http: //www.granitedigital.com), which feature an LCD panel that displays the SMART status of the disk housed inside of it. This gives their enclosures quite an advantage over others, considering that OS X doesn't have the ability to get past the bridgeboard to read the SMART status of the ATA drive inside. Look for other enclosure makers to follow suit in the near future.

Smart, And Its Mirror, Trams

Ok, I'm not talking about Disney World here, and the super long trains of trams that shuttle folks back and forth from the Magic Kingdom to the parking lots. I am talking about RAID Mirrors on OS X, however. One of the problems many admins have complained about with OS X and OS X server, is the lack of notification that a RAID Level 1 (mirror) is going to go south. The upside is that if workstation or server in question does have a mirror, the other drive will continue to function until it's replaced, and the mirror's rebuilt. That's why the word SMART is mirrored above (ha).

Setting up a RAID Level 1 (mirror set) is easy in OS X. After booting off the installation CD for OS X or OS X Server, drag the disks you want to mirror into the RAID window in Disk Utility and choose mirror, then create the mirror set. The result will be a RAID disk that appears as two stacked hard drives in drive and volume list on the left hand side.



Mirror Raid Level 1 in Disk Utility

Of course, diskutil is available for those who want to create, repair, check, or destroy a mirror set from the

terminal. For example, even though SMART status might not be available for SCSI disks, it's easy to script notification for a SCSI mirror, and any RAID 1 mirror for that matter, all based on output from diskutil.

So what I'm looking for in the script below is simply an OK from both disks that are members or "slices" of the RAID

```
host2:~ mostadmin$ diskutil checkRAID disk3
RAID SETS
---------

Name:              boot
Unique ID:         bootf5b49d82471e11d98b06003065be09be
Type:              Mirror
Status:            Running
Device Node:       disk3
-------------------------------------------------------------
#         Device Node                Status
-------------------------------------------------------------
0         disk1                      OK
1         disk2                      OK
-------------------------------------------------------------
```

Mirror set, note how similar it is to the script that checks the SMART status of a single hard drive:

```
#!/bin/sh
## This script is designed to get the status of a mirror set and send
an email notice on fail
## to test, change the "good=2" below to "good=1" and you should
receive a warning email
## Dean Shavit, MOST Training & Consulting
dean@macworkshops.com http://www.macworkshops.com
## This is for servers using software RAID mirror sets only
##
## Step 1: Define a variable for a functional raid by counting the
number of good disks
status=`diskutil checkraid disk3|grep -c OK`
## Step 2: Define a number for comparison against a failed raid
good=2
## Step 3: Define the warning message for the body of the email
warning="houston we have a problem!"
## Step 4: Define a variable for the computer name
box=`/usr/sbin/scutil --get ComputerName`
## Step 5: Define a variable - email address of person to notify
admin="dean@macworkshops.com"
## Step 6: compare current status with good status, if a match, echo,
if not, notify
if [ $status == $good ]; then
echo $status
else echo From: $box- $warning!!! > /tmp/houston.txt| mail -
s "RAID Alert Report" $admin < /tmp/houston.txt
fi
done
```

So, with both the SMART script running as a cron job, along with the RAID script, it's easy to get an advance warning of an impeding single disk failure, or when a RAID Level 1 has a drive fail or go out of sync. It's tougher with or SCSI drives, but if they're used in a RAID Level 1 configuration, the script above can provide the same advance warning that it can for SCSI disks under OS X or OS X Server.

Maxwell Smart

Recently, I was onsite for a couple of months helping a customer upgrade two hundred Macs to OS X. Many of the machines running OS 9 showed signs of hard drive failure, such as frequent boot failures, corrupt directories, slow time to spin up and metallic whining sounds. One day, a department manager's hard drive failed, taking several years of email with it. We sent the drive into DriveSavers (http://www.drivesavers.com) for hardware data recovery, but no dice, the mechanism was too far gone, too much damage done to recover what was needed. It caused quite a stir in the IT department, and a discussion about early warning of hard drive failures. An interesting fact about this customer was that they had several UNIX/LINUX experts on staff who were very interested in learning more about OS X, but interestingly enough, tended to treat an OS X workstation as they would a Linux server. They'd try this command or that command and comment that OS X was very different. I shared my SMART alert script with them, and they found SMARTReporter on Versiontracker. However, they wanted to be able to have a constant real-time log, on each Mac OS X box, of SMART status and more sophisticated scripts that would report back attribute information such as temperature or reallocated (bad) sectors. It was time to go hunting for open-source tools.

The first stop was at the Maxwell web site, http://maxwell.sourceforge.net. Maxwell's a tiny command-line program that queries a hard drive for SMART information and can either report a pass/about to fail condition, just like Disk Utility, or a more comprehensive report of SMART attributes. The source code for Maxwell is less than 100k in size, and is such a simple program, compiling it doesn't require configuration first.

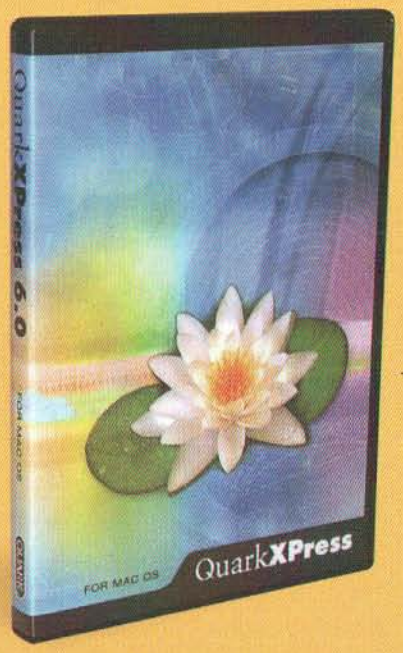
1. To install Maxwell, download the source code from SourceForge, then unpack the tarball into a folder on the Desktop. Open a Terminal window, then navigate to that folder:
2. Issue the following commands as in the example below:

```
[minime:~/Desktop/maxwell-0.5.1] dean% sudo make install
cc    -c -o maxwell.o maxwell.c
cc    -o maxwell -framework IOKit -framework CoreFoundation
                 maxwell.o
/usr/bin/install -d -m 755 /usr/local/doc/maxwell
/usr/bin/install -m 644 LICENSE
/usr/local/doc/maxwell/LICENSE
/usr/bin/install -m 644 README /usr/local/doc/maxwell/README
/usr/bin/install -d -m 755 /usr/local/bin
/usr/bin/install -m 755 maxwell /usr/local/bin/maxwell
/usr/bin/install -d -m 755 /usr/local/man/man8
/usr/bin/install -m 644 maxwell.8
/usr/local/man/man8/maxwell.8
[minime:~/Desktop/maxwell-0.5.1] dean%
```

Once installed Maxwell can be run to get a simple pass/fail status by simply invoking it at the command line:

Mac OS® X compatible

QuarkVista™
image editing



OpenType
fonts from
Linotype*

Direct PDF export

QuarkXClusive™
variable data printing

Multiple levels
of undo

The joy of six.

QuarkXPress 6

QuarkXPress® 6 has arrived. Take it out for a spin at www.quark.com/demo6.
Show it what you can do. It'll return the favor.

```
[minime:~] dean% maxwell
Device: TOSHIBA MK8026GAX                    Reported PASS status

Or, for a more complete SMART report, it can be run with the -r switch:

[minime:~] dean% maxwell -r
BSD Path:/dev/disk0
Serial:              54IK0355T
Model: TOSHIBA MK8026GAX
Firmware: PA002B
Device supports S.M.A.R.T. operations
SMART self-test supported
SMART error logging supported
S.M.A.R.T. operations are enabled
SMART self-test enabled
SMART error logging enabled
Off line collection status is 0
Time to complete Off line Data collection: 3 hours, 20 minutes
Status is GOOD
      TEST                                          THRSH  VALUE  STATUS   RAW
-------------------------------------------------------------------------------
(  1) Raw Read Error Rate                           50     100    0x0b00   0
(  2) Throughput Performance                        50     100    0x0500   0
(  3) Spin Up Time                                  1      100    0x2700   1417
(  4) Start/Stop Count                              0      100    0x3200   430
(  5) Reallocated Sector Count                      50     100    0x3300   0
(  7) Seek Error Rate                               50     100    0x0b00   0
(  8) Seek Time Performance                         50     100    0x0500   0
(  9) Power-On Hours Count **                       0      94     0x3200   2540
( 10) Spin Retry Count                              30     108    0x3300   0
( 12) Device Power Cycle Count                      0      100    0x3200   335
(192) Power Off Retract Count                       0      100    0x3200   2
(193) Load/Unload Cycle Count                       0      78     0x3200   228634
(194) Device Temperature                            0      100    0x2200   223339413546
(196) Reallocation Event Count                      0      100    0x3200   0
(197) Current Pending Sector Count                  0      100    0x3200   0
(198) Off-Line Scan Uncorrectable Sector Count      0      100    0x3000   0
(199) Ultra DMA CRC Error Count                     0      200    0x3200   0
(220) Unknown                                       0      100    0x0200   184
(222) Unknown                                       0      98     0x3200   977
(223) Unknown                                       0      100    0x3200   0
(224) Unknown                                       0      100    0x2200   0
(226) Unknown                                       0      100    0x2600   241
(240) Unknown                                       1      100    0x0100   0
-------------------------------------------------------------------------------
Device temperature is 30 degrees centigrade
-------------------------------------------------------------------------------
The names listed above may not actually be correct for each test value. The real
values printed may make no sense whatever and the temperature may be some crazy
value. Different device manufacturers do things differently :(.
-------------------------------------------------------------------------------
Device: TOSHIBA MK8026GAX                    Reported PASS status
```

With this much information at hand, it's quite a simple matter to filter this output so that, say, the Reallocated Sector Count test had a RAW value other than zero, then it could trigger an alert email, or if the temperature reaches a certain level, or the spin retry count is greater than zero. The man page for Maxwell also suggests that it was designed precisely for scripting purposes and for integration with cron. While Maxwell is certainly a notch above the simplistic pass/fail test of diskutil, it isn't exactly the comprehensive, enterprise-wide type of solution that my customer was looking for, either. They wanted something more integrated into the operating system, a tool that would keep a log, and send that log back to a central syslog server, where trends in hard disk wear and tear could be monitored over a period of time.

It's A Smart World

"It's a world of laughter, world of cheer, it's a world of hope, and a world of fear." Sound familiar? I can almost hear the little mechanical people singing! Our company currently keeps a few servers at Equinix, a state-of-the-art colocation facility near downtown Chicago. With giant steel doors, handprint readers, bag checks, and multiple security stations, it's not hard to imagine at times you're entering the back room of a Disney World exhibit filled with trade secrets. It's an interesting side note, that over the years, I've seen more and more Xserves and Xserve RAIDs populating the cabinets at Equinix. I'm sure many a pager would go off if one or more of those Xserve RAIDs had a drive failure, or if one of the internal disks inside an Xserve died, triggering an alert by the Server Monitor Application.

But now the Mac Mini, Apple's little darling unveiled at MacWorld, seems to have caught on as a the "server for the rest of us." There's even a hosting service for Mac Minis, called "Macminicolo," http://www.macminicolo.com, that allows Mac owners to "park" their Minis in a cabinet for as little as $29.95 per month. While a colocated Xserve with a dedicated 1U space in a rack at a facility goes for the street price of about $150 per month, including one megabit of bandwidth (200 gb of transfer per month), there's absolutely no doubt that the proprietor of Macminicolo can certain cram a whole lot more Mac Minis into the same cubic space an Xserve would occupy.

Rack of Xserves, Rack of Minis

On a side note, the success of (and need for) of such a colocation facility where "everyone can have their own server," shines a glaring spotlight on one of the great weaknesses of Apple's OS X Server software: the lack of virtualization. In the Linux world (on IBM and Red Hat and Sun solutions), virtualization of admin tools and even the actual processors are a standard way of allowing companies to "rent" the resources of the server, without having to actually "own the box." Whether it's only being able to see your users when you log into a virtualized server, or your file system, or on higher end solutions, your virtual server running on a Logical Partition of the server's resources (LPAR), OS X Server's Admin Tools have absolutely no virtualization capabilities as of this writing. It's going to be quite a while before OS X would allow me to log into an eight or sixteen processor Xserve and administer my LPAR, which would appear a single, dual, or four processor server, with my own environment, file system, list of users and server daemons. Today, Apple's solutions seem to focus on miniaturization, not virtualization.

Asides from the obvious management dilemmas that would face the sysadmins of MacMinicolo, where Mac Minis are smushed into a rack three across and three deep with their power supplies and cooling fans. It seems that the system administrators at Macminicolo have found a cool way to monitor their farm of Mac Minis:

> InsideOut monitoring was developed to monitor mac a [sic] based server from the inside out. A headless application that is placed in the startup folder of any OSX machine contacts a master logging server. We monitor factors such as network availability, Hard Drive space, network traffic moved, temperature and the up/down status of specified applications. Customers subscribing to this service can choose to receive email notification if limits or events trigger warnings.

This sounds to me very much like the capabilities of the open-source monitoring software Nagios (http://www.nagios.org), which can log all of the above events and present reports, graphs, and network maps in a web page. Although temperature, application stats, network activity and free hard drive space are all good things to monitor, I would have to make the suggestion that in a situation with multiple Mini Macs with laptop hard drives stuffed like muffins into a rack, SMART status should be the number one indicator monitored, because should a hard drive fail, there'll not only be an unhappy customer who will have lost all of their data if they don't have an offsite backup, but there'll also be an unhappy technician scrambling to pull out a single Mini and replace the hard drive as quickly as possible. If a SMART monitor could give advance warning of a failure, that would make the whole process much easier on everyone.

So, in addition to an enterprise or medium-sized business needing to monitor the SMART status on lots of hard drives in client workstations, it seems that Macminicolo would be another good example of where it would work quite well. However, the solutions I've looked at so far, like diskutil and the script I wrote, SMARTReporter, and Maxwell, are all missing an important capability, none of them have the ability to write a log of SMART events either on the local machine, or to a syslog server on the network, which of course, was the same thing my customer wanted for their Macs.

The Smart Test Of All

Much of the really great open-source software for OS X seems to originate from the Linux community, and such is the case with the smartmontools, http://smartmontools.sourceforge.net, which by all accounts is the top tier solution for SMART monitoring and logging in a non-commercial distribution. Bruce Allen, maintainer of the smartmontools project, explains the origins of the package in an article that appeared in the Linux Journal:

> By profession I am a physicist. My research group runs a large computing cluster with 300 nodes and 600 disk drives, on which more than 50TB of physics data are stored. I became interested in SMART several years ago when I realized it could help reduce downtime and keep our cluster operating more reliably. For about a year I have been maintaining an open-source package called smartmontools. . .

In July 2004, Geoff Keating ported the smartmontools to OS X for ATA drive support only, (though the Linux version includes support for ATA/ATAPI-3 to -7 disks and SCSI disk and tape devices) bringing a sophisticated and flexible SMART monitoring suite to OS X for the first time since Granite Digital discontinued their commercial SMARTVue™ software nearly two years ago. Smartmontools consists of two main components, a command, smartctl, and a daemon, smartd. While smartctl functions like an improved version of Maxwell with various switches to control the verbosity and depth of the reports and self-tests, it's smartd that really takes the monitoring to a real-time level, where any changes in SMART status, not just errors, are logged to a specified syslog facility.

To install the smartmontools from the source code, make sure you have the latest Xcode tools (Apple Developer Tools) installed,

then download the source tarball from http://smartmontools.sourceforge.net, and expand it to a folder on your Desktop, there's many download packages available, but the one we're looking for is called: smartmontools-5.32.tar.gz. Open the Terminal, navigate inside the folder and do the following:

```
minime:~/Desktop/smartmontools-5.32 dean$ ./configure
```

You'll see a long list of checking for (various components). When the configure process is finished, type the following:

```
minime:~/Desktop/smartmontools-5.32 dean$ sudo make install
```

After a page or two of build feedback, smartmontools is now installed on your system. To start smartd, type the following command:

```
minime:~/dean$ /usr/local/etc/rc.d/init.d/smartd start
```

To run smartd automatically on startup, type the following:

```
minime:~/dean$ sudo ditto /usr/local/etc/rc.d/init.d/
/Library/StartupItems/SMART/
```

Now you'll need to add a line to the end of your /etc/hostconfig file that reads:

```
SMART=-YES-
```

That will allow the SMART StartupItem to run every time your Mac boots, and you'll also see the reassuring message "Starting SMART disk monitoring" as the daemon loads before the Login Window appears. Now that smartd is running as part of the OS X startup process, the next step's to configure logging:

```
minime:~/dean$ sudo touch /var/log/smartd.log
```

This creates an empty logfile to contain the smartd output. The next step is add the line below to the /etc/syslogd.conf file so that smartd knows where to write that output, in this case using the free local3 log facility:

```
local3.*                    /var/log/smartd.log
```

or, if you'd like to send the log to a remote syslog server, then

```
local3.*                    @my.syslogserver.domain
```

and, if you'd like to make sure that the smartd.log file is rotated weekly, modify the /etc/periodic/weekly/500.weekly file, so that the following loop statement has smartd.log in the list of rotated log files.

```
for i in ftp.log lookupd.log lpr.log mail.log netinfo.log
hwmond.log ipfw.log smartd.log; do
```

Test it with:

```
minime:/etc/periodic/weekly dean$ sudo 500.weekly
```

You should see the following line appear:

```
Rotating log files: ftp.log lookupd.log lpr.log mail.log
netinfo.log ipfw.log smartd.log
```

Lastly, you'll need to modify the /usr/local/bin/etc/smartd.conf file to specify which drives you'd like to monitor, and which tests you'd like to run, as well as an admin notification email address. Remember, in OS X the startup disk is almost always disk0, if the computer has a single physical drive, when the /dev/hda placeholders in the smartd.conf file are meant for other UNIX systems, not OS X.

First, open up the /usr/local/etc/smartd.conf file in your favorite editor. Luckily, the smartd.conf file has an excellent synopsis of what each command and directive accomplishes, so it's pretty self-explanatory:

```
minime dean$ sudo pico /usr/local/etc/smartd.conf
```

Find the line that reads:

```
DEVICESCAN
```

And edit it so that it reads

```
#DEVICESCAN
```

This will tell the configuration file to explicitly use the device you specify, rather than the one it finds on its own.

Find the line that reads:

```
#/dev/hda -a -o on -S on -s (S/../../.02|L/../../6/03)
```

And edit it so that it reads

```
/dev/disk0 -a -o on -S on -s (S/../../.02|L/../../6/03)
```

This command will perform short and long self-tests and report full SMART status results to the smartd.log file on a schedule.

Find the line that reads:

```
#/dev/hdc -H -m admin@example.com
```

And edit it so that it reads

```
/dev/disk0 -H -m dean@macworkshops.com
```

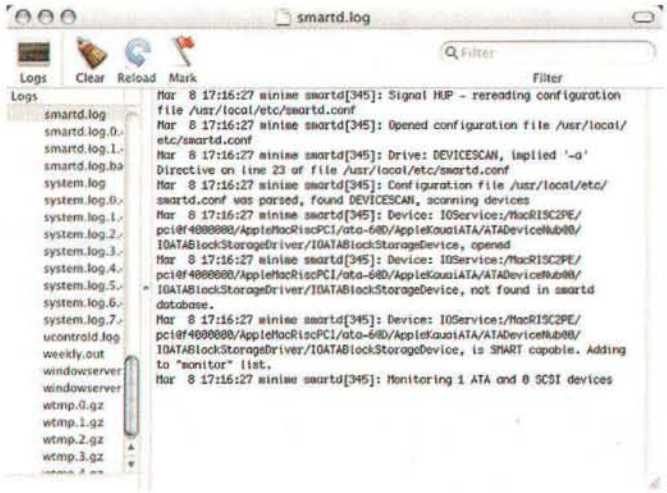
This test will notify you by email if the simple health check (pass/about to fail) shows that the disk is in trouble. Now, restart the smartd process with the following command so that smartd will re-read its configuration file:

```
minime:~/dean$ sudo killall -HUP smartd
```

Now the smartd.log is visible in the Console application located in the /Applications/utilities folder, and we can read the events we just kicked off. Now your Mac has a real-time log of SMART events and a notification that is emailed immediately when the failure occurs, rather than waiting for cron to do its thing.

With log analysis tools such as Oak (http://www.ktools.org/oak.html), smartd logs can be analyzed to project trends of failure that might occur in waves, due to defective hard drives. Remember the Quantum Bigfoot 5 _ inch hard drives from the late 90s? They never were shipped in a Mac, but at the time Compaq put them in tens of thousands of their Deskpro workstations, and nearly every single disk drive had to be replaced. A tool like smartd can help prepare companies for a run of defective hard drives that might die, en masse. Other features of smartd include

a database of hard drive SMART attributes (not nearly complete, but you can submit a device report from your hard drive for submission to the database by using smartctl –a to and emailing it). The other "half" of the smartmon tools is the smartctl command, which can be queried through cron scripts or directly from the command line, much like Maxwell.



Console

Everybody Now . . .

I realize that downloading, compiling, and configuring the smartmontools isn't for everyone, but it certainly is a rewarding journey for those administrators who want piece of mind and a logfile, at least until the storm hits. . .but even for consultants who support home users the smartmontools or even a simple script or SMARTReporter can be a real-life saver. Wouldn't it be nice to call a customer and say, "Hey, I'm going to come over with a brand-new hard disk, backup your data, and replace your drive, because it's going to fail . . ." Or better yet, "Backup your data, now, before it's too late!" Or even, "It seems like those Hitachi hard drives are dropping sectors like crazy, maybe we'd better turn up the air conditioners." So, for those who'd like to get started quickly with the smartmontools, I've whipped up a Installer package that puts the smartmontools on your Mac, as well as the requisite StartupItem. However, you'd still need to configure the /etc/hostconfig, /etc/syslog.conf, /usr/local/etc/smartd.conf files and create a smartd.log file yourself. The installer is available along with the scripts in this article at http://www.themachelpdesk.com. Now, I'd like everybody to join me ". . . There's so much that we share, that it's time we're aware, it's a SMART world after all. . ."

In Next Month's "Source Hound"

Steve Jobs, in his keynote address at MacWorld, repeated the following mantra over and over "H.264, H.264, H.264." Sure, H.264 support is with QuickTime 7, but if you want it now, it's available today, not from Apple, but from the deepest trenches of the open-source world, the realm of the Coneheads and the video rogues, where encoders are encoders and QuickTime is QuickTime, and never the twain shall meet, or will they?

About The Author

Dean Shavit is an ACSA (Apple Certified System Administrator) who loves to use a Mac, but hates paying for software. Since he's not into breaking the law, his most common response to any cool solution is: "Does that cost money?" If it does, you can bet he's on the hunt for an Open-Source or freeware alternative. Besides surfing for hours, following the scent of great source code, he's a partner at MOST Training & Consulting in Chicago, where he trains system administrators in OS X and OS X Server, conducts large-scale Mac Deployment and Upgrade projects, and writes for his own website, http://www.themachelpdesk.com. If you have questions or comments you can contact him: dean@macworkshops.com.

Focus Review

By Lesa Snider

Near-Time's Flow™

Peer-to-peer content and knowledge management for Panther

What do you get when you cross iChat, Safari, an RSS news reader, Microsoft Word, Adobe GoLive, and a handful of other popular productivity tools? You get Near-Time Flow™; an extremely fascinating and powerful content/knowledge management, collaborative workflow solution for Panther.

Released this past June post-WWDC, Flow is virtually impossible to categorize as a single piece of software. After spending some quality time with it, though, you come to understand that it is part WYSIWYG web editor, RSS aggregator, word processor, outline tool, email client, and then some. This unique, hypertext-based software is targeted to both individuals and work groups of any size. It's an authoring tool, a gathering tool, an organizational tool, a collaboration tool... it should have a big red S emblazoned on its box!

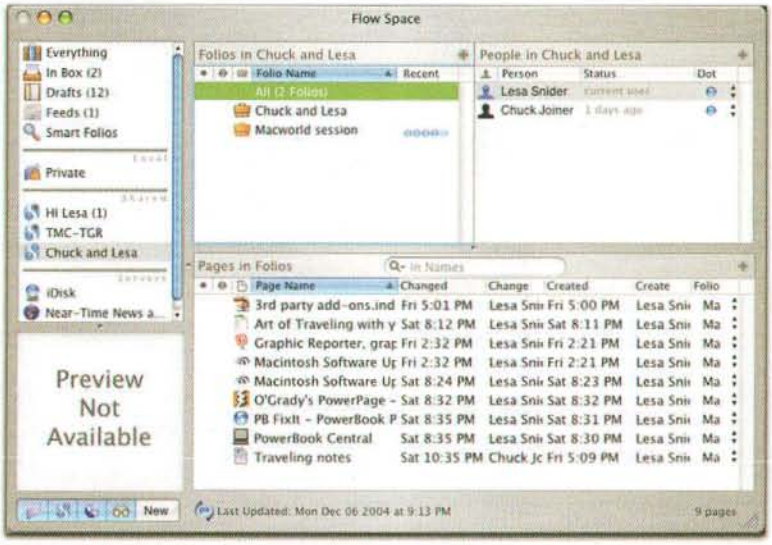


Figure 1. Your Flow Space lists Folios, Shared Spaces, other Flow users within those spaces, and available servers.

Creating, gathering, and organizing

Life within Flow happens in two windows simultaneously, metaphorically dubbed Flow Space and Folio. From within your Flow Space you can create multiple Folios specific to your projects that can be shared or kept private. Once you've created a Folio, you can add any type of file by dragging it into the Pages pane of the Folio window where everything appears in an orderly list that can be arranged to suit your needs. Add RSS feeds, web pages, images, and other documents and enjoy the bliss of managing, viewing, and commenting on them all in one convenient place, within a single interface. It even dutifully alerts you when an updated RSS feed is available by placing a blue dot beside it. To gain an appreciation of how easy and powerful the Folio metaphor is, visit Near-Time's web site and watch the QuickTime movies illustrating several of Flow's key features.

You can also create new content in either text or HTML format, yielding a native Flow document. I found the Ruler option to be quite handy in this process, though text formatting is handled via a drop-down window similar to—and as inconvenient as—that found in Keynote. You can link content pages together by simply dragging them from the Pages list to the Content area of another. Add web pages

and RSS feeds by dragging the URL straight into your Folio from a browser's address bar.

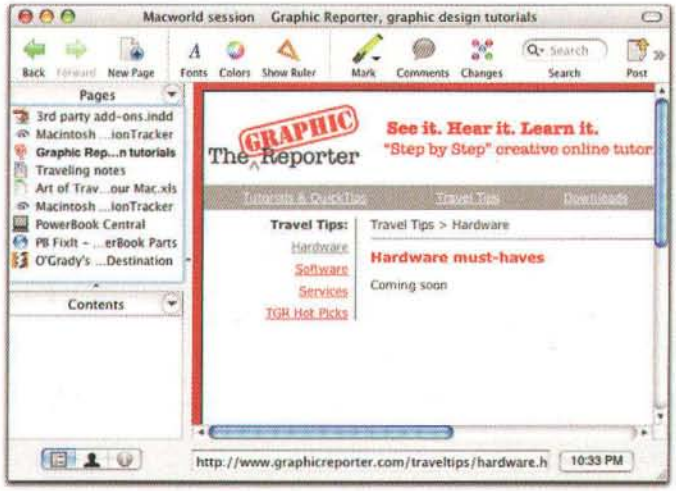


Figure 2. From within a Folio you can view content assets such as live URLs and more.

Sharing and publishing

Not only does Flow allow you to gather and organize all of your project documents, it gives you the ability to share and publish them out to others as well; Flow users or not. These options include:

- sharing Folios through Rendezvous or other internal network
- sharing Folios through Real-Time's server with other Flow users (available for free for one month after purchase, then $29.95 per year thereafter)
- publishing Folios in web site form to an iDisk or other server

Once a Folio is shared, all files—even those of external origin—are transferred back and forth between the Flow users within a Shared Space. The touch of a button activates a manual sync, though Flow checks periodically and notifies you of any content updates with the aforementioned appearance of blue dots. Unfortunately, a major flaw showed up during testing when Flow became confused as to who had edited the last version of an Excel spreadsheet, and caused both my colleague and I to lose changes the other had made. Similar losses occurred with comments made on an Adobe InDesign file. The sharing process worked flawlessly with native Flow documents, RSS feeds, and web pages.

Flow supports version tracking on all content through a convenient date/time stamp bar at the bottom of each Folio document. Not only can you see when a document was edited, marked, or commented upon, but you can also see who made the change. I was impressed to find that Flow also accounts for differences in time zones between users in a Shared Space while testing its collaboration features with my colleague on the East coast. The Folio workspace itself would benefit from more standard controls for text editing (bold, italic, etc.) instead of the drop-down menu, as well as some visual cue as to what mode one is in, especially when invoking changes. I kept forgetting to click the Make Changes button at the bottom of my Folio window while trying to add comments to a document.

The publishing aspect of Flow is effective, allowing the creation of visually pleasing project web sites with a single button-press, enabling the sharing of Folios between those who use Flow and those who do not. We experienced a few problems when publishing a Folio to an iDisk: native Flow documents, linked URLs and RSS feeds uploaded flawlessly; however, attached files (Excel spreadsheets, InDesign documents, etc.) were included only part of the time—sometimes they would be uploaded to the server and sometimes not. To Flow's credit, it gleans iDisk data from your System Preferences; however, you must manually enter your iDisk password. Flow failed to snag it on both my and my review partner's system.

Smart Folios, blogging, and searching

Another impressive Flow feature is the Smart Folio option. By creating rules you can automatically and dynamically aggregate information from various sources. The concept is similar to Smart Playlists in iTunes and is at least as useful, especially as a project expands both in size, and number of team members involved.

If you are interested in the ever-expanding world of blogging, you will find Flow's options to create content and publish to Moveable Type, TypePad, Wordpress, or Blogger extremely useful. Simple, one-button uploads to your blog may be a compelling reason to give Flow a try.

Flow's sophisticated Search feature performed flawlessly. The best way to appreciate it is to open an existing Flow document (e.g. the included User's Guide), and then query it using a search panel that

looks suspiciously like the one in Panther's Finder. It then creates a brand new page with all the results listed, highlighting the search terms as hyperlinks back to the appropriate section of the documentation. One can easily imagine how publishing this page to other Flow users would facilitate work on specific aspects of any project.

Final word

As a single, individual project management and authoring tool, Flow largely lives up to its claims. You can easily and effectively collect and manage the assets of a particular project within the Flow Space and Folio metaphors. As long as you're working with assets that Flow understands and can edit, it is also an effective collaborative tool, making the sharing of project documents between development teams and departments much easier than via email.

Flow's kryptonite, if you will, seems to be only when you mix in files that the program does not understand and must handle as attachments—that's where it became less reliable. The program is priced to be accessible to anyone who has serious collaborative needs, and is powerful enough to be at the top of the line of a new category of productivity software, once a few of the inconsistencies are worked out. To get a full appreciation for what Flow can do, you will need to climb a significant learning curve, though the trip is worth it to access the software's power for your projects.

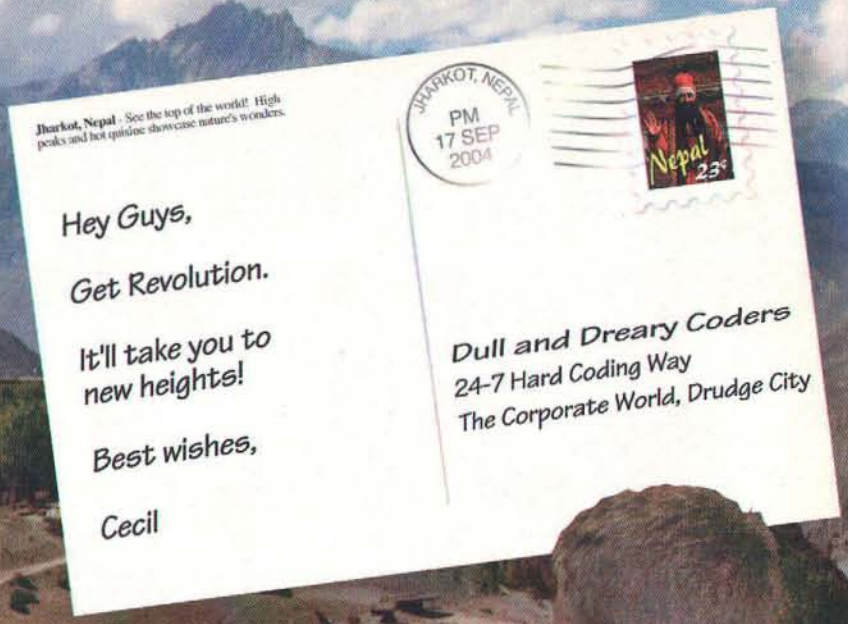
You can try it for 30 days for free by downloading the trial version from the company's web site at http://www.near-time.com. The cost for a single seat is $99.90, and a ten user license is $859.95, a respectable value to be certain.

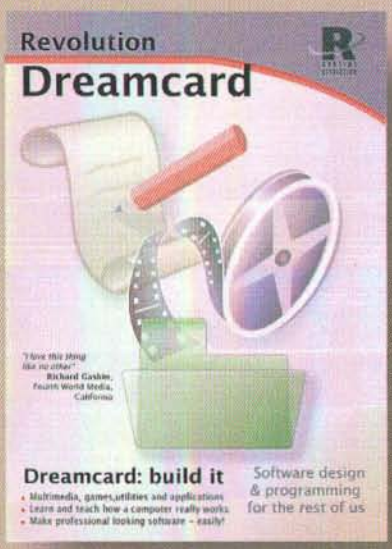
About The Author

Lesa Snider is the owner of Flying Fingers (www.flyfingers.com), a creative new media firm specializing in web site visibility. She provides private and corporate training and consulting on web site marketing techniques, interface design, and search engine optimization, as well as a variety of design disciplines.

www.runrev.com

Some things ... Are too good to leave behind. Introducing





Introducing

Dreamcard: build it

Software design & programming for the rest of us...

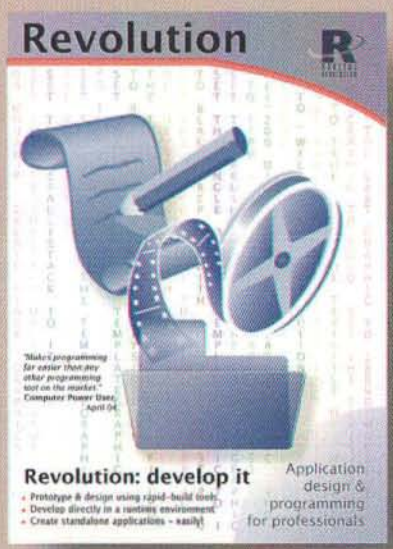
- Make multimedia, games and applications
- Learn and teach how a computer really works
- Make professional-looking software - easily!

Just released - 2.5

Revolution: develop it

Application design & programming for professionals...

- Prototype & design using rapid-build tools
- Develop directly in a runtime environment
- Create multi-platform standalone applications - easily!



"Makes programming far easier than any other programming tool on the market"
Computer Power User, April 04

RUNTIME REVOLUTION
User-Centric Development

Runtime Revolution • 15-19 York Place • Edinburgh EH1 3EB • Scotland • UK
Phone +44 (0) 870 747 1165 • Fax +44 (0) 845 458 8487 • www.runrev.com • Email info@runrev.com

Cron and Crontab In-Depth

Making sense of 30 4 1,15 5

By Josh Wisenbaker

What's this all about?

One of the most essential, and misunderstood, UNIX admin skills is being able to effectively use the cron daemon by maintaining your crontab files. The big problem is that the format of the crontab files can be very hard to grasp at first. Full of bizarre looking statements such as "30 4 1,15 5", the crontab scheduling statements can put off many users before they begin. By the end of this article it is my hope that you will not only have a firm grasp of the crontab format but also have some good ideas of what you can do with cron.

Cron

O.K. here we go. Cron is the name of the program that runs scheduled tasks on your Mac. For example you can use cron to call a command to upload files to a web server, fetchmail from a remote server, roll and archive log files, or clear cache files. In general, any non-interactive task can be easily automated with cron.

Like most things on Mac OS X, cron is almost, but not quite, like cron on other UNIX systems. The main difference is that it is started by the /System/Library/StartupItems/Cron startup item rather than the /etc/rc boot script. Other than that, you can treat cron on Mac OS X the same as you would on any other FreeBSD or Linux type system.

When cron runs a command or script it can output the results to a log, e-mail them to a user, or just discard them and not bother you with the details.

It does all this by reading a file, called a crontab, once a minute to see what needs to be run and when. There can be several crontab files on the system. By default there is only the system crontab located at /etc/crontab. This file contains things like when periodic should run to perform the daily, weekly, and monthly scripts. Each user on the system can also have a crontab file so that non-admin users may schedule tasks as well and these are stored in /var/cron/tabs/<username>. It should be noted however that the format of the system crontab is slightly different than that of the user crontabs and the methods used to edit them are very different.

Crontab Demystified

The Basics

Before we take a look at how to edit the crontab files let's take a moment to get the basic formatting figured out. That way when you open /etc/crontab for the first time it will make sense to you. Although it seems mystical at first, the format for a crontab is actually very simple. It breaks down like this:

```
minute hour day-of-the-month month day-of-the-week
command
```

The values are separated by white space and the final value, command, may contain spaces of it's own. Let's look at the five values that make up the time to run section...

For the any of the settings you can use numbers starting at 0 and you can optionally use names for the month and day-of-the-week settings thanks to Mac OS X's FreeBSD roots. (Really this is due to the fact that FreeBSD uses an enhanced version of cron, written by Paul Vixie.) When using the names of months or days just use the first three letters in a case-insensitive fashion. A "*" is a wildcard that means first-last and matches anything. So let's say we want to schedule a task to run on the hour, every hour.

```
0 * * * * command
```

Huh? How does that work? Easy. The 0 minute is the top of the hour. The wildcards match all other possibilities. So it breaks down to minute 0 of every hour, of every day, of every month, every day of the week. Let's look at another one. Let's say you want to run something every Sunday at 12:15 am.

```
15 0 * * sun command
```

So you are telling it run on minute 15, of the 0 hour (midnight), every day of the month, every month, on Sunday. Notice here that you can use 'sun' for Sunday. You could also use 0 or 7 for Sunday if so inclined. Let's take a look at one more basic example. Pretend we want to run a task at the beginning of each New Year.

```
1 0 1 1 * command
```

Cron will interpret this as January 1 at 12:01 am every year. Just because it is a repeating task it doesn't mean that it needs to repeat very often. If you want to start spacing your jobs out in intervals larger than a year though you will need to take care of that in the command or script that you call from cron.

Step Values, Ranges, and Multiples

If we were restricted to simple schedules like the ones just given, cron would not be very flexible. Luckily we can get a bit more creative with our crontab entries. Say that you've decided that you need a task to run at 12:15 am on both Sunday and Tuesday nights. You could make multiple crontab entries or use a value list.

```
15 0 * * 0,2 command
```

Will accomplish our desired scheduling with only one entry. Notice here that we used the numerical values for the days although "sun,tue" would work as well. A second option at our disposal is the step value.

If you wanted something to run every two hours you could specify "0,2,4,6,8,10,12,14,16,18,20,22" for the second field but that is really messy. Luckily we can use a step value to make this easier and arrive at the same result with "*/2". So let's say you want to make something that runs every other month on Sundays at 4:32 am...

```
32 4 * */2 sun command
```

You can also use a range in place of a list. Rather than type "9,10,11,12,13,14,15,16", just use "9-16".

```
0 9-16 * * * command
```

This will run our command every day, on the hour, from 9:00 am to 5:00 pm. Since Mac OS X is a FreeBSD based system we can combine ranges, lists, and step values in a single crontab entry. Building on the last example:

```
0 0-3,9-16/2 * * * command
```

Will run our command every day, on the hour, from 12:00 am to 3:00 am and again every other hour from 9:00 am to 5:00 pm. Systems that do not use the Vixie cron would choke on this.

More BSD Family Perks

Since Mac OS X is FreeBSD based we can use some handy shortcuts. Remember our first example, '0 * * * *' that ran a task every hour? Well, you could also just say

```
@hourly command
```

Isn't that nice? You can also use @reboot (Once on startup), @yearly (0 0 1 1 *), @annually (same as @yearly), @monthly (0 0 1 * *), @weekly (0 0 * * 0), @daily (0 0 * * *), and @midnight (Same as @daily).

Another set of options is available to us as well in the form of environment variables. You can use the SHELL variable to override the default setting of /bin/sh. Usage is simple:

```
SHELL=/bin/bash
```

Just place this, or any of the following variables, at the beginning of your crontab. While the SHELL variable is a bit obscure to many administrators, the PATH, MAILTO, and HOME are a bit more familiar. By default cron will just use the path defined for the owner of the crontab but you can use the PATH variable to override it...

```
PATH=/etc:/bin:/sbin:/usr/bin:/usr/sbin
```

You can use the HOME variable in a similar fashion.

```
HOME=/var/log
```

By far the most useful environment variable we can define is MAILTO and it has a few usages. By default cron will e-mail the output and results of commands to the owner of the crontab file. We can modify this behavior in two different ways:

```
MAILTO="josh"
```

will redirect all mail for the crontab to the user named "josh" and

```
MAILTO=""
```

will suppress any mail from the crontab at all. You may find this option useful for a crontab file whose commands that are logged, but do not need to be mailed to anyone.

Command time

OK, now that we know how to schedule something what can we do? Well, anything you want! You can run single commands or scripts and with advanced Mac OS X features like osascript you can even run AppleScripts to drive things like Photoshop. There are only a few things to keep in mind when typing your commands with the main one being that it must be all on one line. If you need to use a newline for your command you can embed a "%" to indicate this. Take a look at the following example.

```
0 0 2 2 * mail -s "B-Day!" josh%Hey!%Happy birthday Josh!%
```

This will produce an e-mail with the appropriate newline returns in it. If you want to actually use the "%" character in your command you will need to escape it with a backslash.

You can also string together multiple commands with pipes and semicolons just as you would in the Terminal. If you wanted to get a daily update from the login accounting systems (Don't confuse simple commands like ac and last with the recently released Solaris BSM auditing tools from Apple.) via e-mail you could use the following entry.

```
0 10 * * * (ac -p; echo; last | tail)| mail -s "Login
Accounting" admin@foo.com
```

Notice that the pipes work just like you would expect them to and the parenthesis are used to group the output of multiple commands into one result. It's also worth noting that the output of the ac and last commands will be sent to admin@foo.com but the report that the command was run will be sent to whoever owns the crontab or a user specified with the MAILTO variable.

So far all of these examples are fairly trivial one-line commands, but that doesn't mean one-liners can't be powerful. If you support designers using Adobe Illustrator then you have run into Adobe font cache issues. Wouldn't it be nice if those nasty things just magically disappeared each night?

```
15 0 * * * find -x / -type f -name 'AdobeFnt*.lst' -exec rm
-f {}\;
```

If you add this to root's crontab or the system crontab then the stupid Adobe font caches vanish every night at 12:15 am.

You can only stretch one command so far though, so you will find times when you need to resort to calling a shell script with cron. Here is a simple script I use to backup my home folder if a particular FireWire drive is connected using psync.

```
#!/bin/bash

if [ -d /Volumes/TinyDrive/Backup ]
then
/usr/local/bin/psync -d /Users/josh\
/Volumes/TinyDrive/Backup/josh
exit 0
else
exit 0
fi
```

I just call this once an hour with cron and my home folder stays backed up safe and sound.

Extra command tidbits

As I mentioned earlier, cron sends an e-mail to the owner of the cron with the results of each job that runs. But what if you don't want any output from a particular command? The easiest way to snuff the results of one command is just to send the output to /dev/null. When I run my backup shell script I send both stdout and stderr to the bit bucket...

```
@hourly /backup.sh 2>&1 /dev/null
```

This way it never bugs me with the results. If I were just to send stdout to /dev/null then it would just send me the errors. Redirecting the output of a specific command is much more selective than using a MAILTO="", as that option will kill all output from the crontab.

There is one more, really cool, Apple added feature. Let's say that I have a sync operation that runs every 15 minutes. Now if it is critical data I probably don't want this to run if the power is out and the system is on the UPS. If you prefix a command with '@AppleNotOnBattery' then cron will not execute the command if not on AC power. So,

```
@AppleNotOnBattery command
```

is all you need.

Editing your crontab files

There are two distinctly different types of crontab files on Mac OS X, the system crontab and user crontab files. They have slightly different formatting, but are edited in very different ways. Let's take a look at the system crontab first.

The system crontab

Located at /etc/crontab, the system crontab is typically used for system maintenance activities such as the periodic command. One of the nice things about the

system crontab is that it is commented and the first thing you should notice is that it has an extra field in the entry lines with a "who" variable between the weekday and command. Since any particular user does not own this crontab you must specify whom the command will have as its owner. Take a look at this example.

```
15 3 * * *  root     periodic daily
15 0 * * *  mailman  /usr/share/mailman/bin/check_perms
```

The first command is the daily run of the periodic command that all Mac OS X computers come with. Notice it runs as root. The second command checks the permissions on the mailman install of a Mac OS X Server. Notice it runs as the mailman user. For security reasons you should keep the amount of tasks performed as root to a minimum. In this case the mailman user has the rights it needs to get the job done, and it owns the files in question, so it makes sense to run this command with that user.

To edit the system crontab you simply open it with your editor of choice. Beware of line wraps though when using tools such as pico. (Since pico is such a popular editor I will point out that running it with a –w flag will minimize line wraps.) Remember that this file is owned by root, so you will need to use sudo to edit it. Simple eh?

User crontab files

One of the nice things about cron is that any user on a system may use it. However, it should be obvious that you don't want regular users messing about in the system crontab. Luckily there is a system provided to get around this issue and each user may have an individual crontab file. The individual user crontab files are kept in /var/cron/tabs/<username> and, unlike the system crontab, they may not be edited directly. Instead you must use the crontab command.

The crontab command has a few basic usages. To edit your personal crontab simply use:

```
crontab -e
```

and your crontab file will open up for editing. When it opens the crontab it does so using the default editor, which unless you have changed it, is vim. (Now some of you will yell at me for it, but vi is a six headed beast to me and I changed my default editor to pico by adding "export EDITOR=pico" to my .profile. The only real requirement of the editor is that it can edit the file in place without unlinking it.) Once the file opens up, just create your entries and save them. That's it! Cron will evaluate all the crontab files on the next minute and execute anything that it finds. No HUPing needed.

You can also use the crontab command to list your existing entries with a –l flag or remove your crontab with a –r flag. If you choose to use the remove flag the system will prompt you for approval before deleting the file.

If you have administrator rights you can edit the crontab files of other users as well. If you remember our mailman example from before, we could just make that entry directly in the mailman user's crontab file by using the –u flag.

```
sudo crontab -u mailman -e
```

Now mailman's crontab will open and you can add jobs to it as you see fit. Note that if you simply used "sudo crontab –e" that it would open root's crontab file. This can get confusing though and I would recommend that you always specify the user when editing a crontab file other than your own.

Security issues

Now that you know that any user can create and use a crontab file your admin senses should be tingling, alerting you to the danger of unrestricted cron jobs. Out of the box there is nothing stopping a user from leaving a time bomb of a command in their crontab file. You might not even realize it is there until it unleashes its havoc at some later date. Imagine having 5000 users in your system that have shell access, but you only need root, bugzilla, and mailman to have access to cron. What can you do? The solution is simple as cron, like many other daemons, has a facility for allow and deny files.

Located at /var/cron/allow and /var/cron/deny, these files provide a simple way to restrict crontab usage. If the allow file exists, users must be listed in the file in order to use crontab. If your system has a deny file but no allow file, then users must not be listed in the deny file in order to use crontab. By default, neither of these files exists. On a basic Mac OS X system this allows unfettered access to the crontab command.

By using an allow file without a deny file you can easily restrict crontab to just a few users, thus making your Mac OS X system much more secure.

Wrapping up

So that's it! Armed with your newfound cron and crontab skills you can begin to automate all sorts of tasks on your Mac OS X systems. Backups, file archiving, software updates, and trash patrol can all be automated. The more you have the system work for you, the less work you need to do for it and isn't that the goal of every sysadmin?

About The Author

During the day, Josh Wisenbaker is the Sr. Systems Engineer for a Macintosh IT company in Winston-Salem, NC. He is better known for his work as half of www.afp548.com. You can reach him at macshome@afp548.com.

MOVABLE TYPE ON PANTHER

NO NEED TO WAIT FOR TIGER TO GET A TOPNOTCH BLOGGING SYSTEM ON MAC OS X SERVER

Welcome

Now, while Tiger Server will give you a blog as part of the standard installation, there are a number of reasons not to wait for that. For one, while I'm quite sure that Apple's blog setup will be functional and work well for most; it may not be a system that does everything you need it to do. (Everything I've read indicates that Tiger's blog setup will be based on Blojsom, a Java blog implementation.)

Secondly, if you are going to use a blog as a production system, you'll want to test Tiger server out thoroughly before unleashing it on your users. Finally, why wait? There are blog systems that you can run on Mac OS X Server today, and we'll take a look at installing and configuring one of them, namely Movable Type, from Six Apart, (http://www.sixapart.com/), creators of Movable Type, the TypePad blogging service, and new owners of LiveJournal, a public blogging community based on Open Source software.

Why Movable Type?

Well, for a number of reasons. One, I'm familiar with it, as my own website, (http://www.bynkii.com/) is a Movable Type, (Movable Type) based site. (No, I'm not an HTML whiz, which is one reason why I like blogs.) It's quite popular in the Mac world, with sites such as NSLog();, and Daring Fireball among the sites using Movable Type. It can run easily on Mac OS X Server, once you get how to install it, and it can use various databases as a back end, such as BerkeleyDB, MySQL, PostgreSQL, and SQLite.

Movable Type uses CSS and other standard web technologies to create its "look" so if you don't like any of the defaults, you can change it to suit your needs without having to worry about what it will do to various browsers, and almost any web design tool can be used with Movable Type.

Movable Type is well documented, and has a solid API that can be used to extend its featureset a number of ways, including plugins. This is not to say that the other systems aren't just as good, but I had to pick one, and I know Movable Type, so, this is the one we look at.

Getting Movable Type

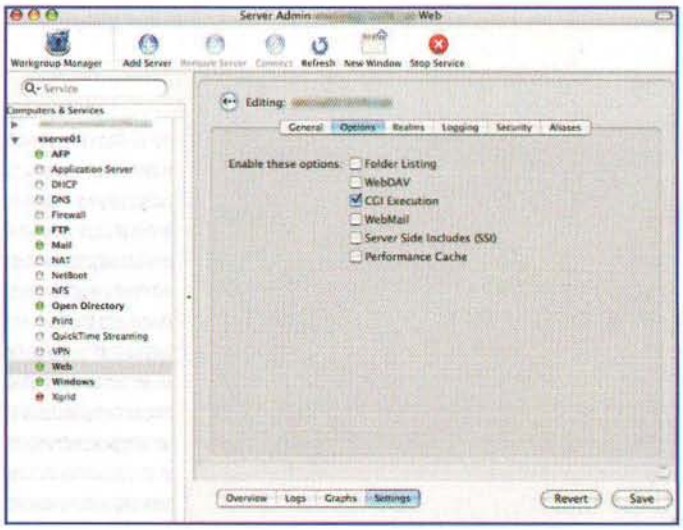
Getting Movable Type can be a little tedious, but there are a number of license configurations that will work for anyone, from free, to not ridiculously expensive, depending on the number of blogs and authors you want. Note that with Movable Type, if you have separate blogs that all fall under the same root domain, (such as those I use for the various sections of Bynkii.com), that's considered a single blog as far as licensing is concerned. Go to http://www.sixapart.com/movabletype/pricing, and pick the configuration that works best for you.

Once you've selected your licensing configuration, download the full install of the latest version of Movable Type, 3.15 as of this article. (You'll have to get a free TypePad ID to get Movable Type, but in my experience, Six Apart has yet to spam me, or email me at all for that matter, and I've been using Movable Type for a little over two years now.) You select the format you want, .tar.gz or .zip, and download it to your Mac, either the server you'll be running it on, or an administration station. Expand the archive, and then immediately navigate into the docs subdirectory, and open mtinstall.html in a browser. This contains the primary installation documentation, and you will want to be quite familiar with it. Almost every problem I've ever had with a Movable Type installation on Mac OS X Server is either due to not paying attention to the directions, or Apple's changes to a standard Apache install.

With all the hoopla about Mac OS X Server 10.4, and all the new tricks it's going to bring as part of the standard installation, I thought I'd take a look at one that's near and dear to my heart, and see about implementing something similar on the current version of Mac OS X Server, namely the Weblog, or "blog" implementation.

Installing Movable Type

Okay, so we've read the instructions, and we're going to set up Movable Type on our Mac OS X box. To keep things simple for this article, we're going to use the BerkelyDB as the back end. This is where things get a little odd thanks to Apple's...unique take on Apache directory setups. First, you want to ensure that Perl is in the right place, since Movable Type makes heavy use of Perl. Open a command line window in terminal and run "`which perl`" The answer should be `/usr/bin/perl`, as that's the normal place that Mac OS X puts it. If you've modified your perl location, follow the Movable Type instructions on changing the CGI files so they know where to look.



Site Settings for Movable Type

We're going to install the Movable Type CGI files into Apple's cgi-bin directory, so we'll want to create an mt-static directory in the main web documents directory. For our example, we'll use /Library/WebServer/Documents/. You can have this directory anywhere along with your actual non – CGI files, but again, we're keeping it simple. We're going to put all the CGI files into the default CGI location, /Library/WebServer/CGI-Executables/.

However, before we do any of this, we want to make sure that the Web services for this site in Mac OS X Server are set correctly. Start up Server Admin, and select the Web settings. Go to the lower Settings tab in Web services, and select the sites tab. Open up the site you're going to use for your Movable Type installation, and under options, enable "CGI Execution" and disable "Performance Cache". You'll also want to go into the Modules tab and make sure that the perl_module and php_modules are enabled, since Movable Type uses both. Save those changes, and exit Server Admin.

In terminal, (you can do all of this in the Finder, or even via SFTP, but in the end, terminal ends up being a more direct and simpler way to set up Movable Type.), cd to /Library/Documents/ and create a directory named *mt-static*. Set the permissions so owner and group have full permissions, everyone else has read and execute only, i.e. `chmod 775 mt-static`. Then copy the following items into this folder, as per the Movable Type installation instructions, (which, since you read them thoroughly before starting, you already know to do this.)

- The mt.js file
- The styles.css file
- The docs folder
- The images folder

Once those are in mt-static, I set the permissions for them as follows, (Note: This is for a server that *only* talks to internal clients. If this server will be exposed to the public Internet, you *will* want to set your permissions more restrictively):

- mt.js to full permissions for owner and group, read and execute only for all others (`chmod 775`)
- styles.css to read & write for owner and group, read only for all others (`chmod 664`)
- The *contents* of the docs folder to read & write for owner and group, read only for all others (`chmod 664`)
- The docs folder itself to full permissions for the owner and group, read and execute only for all others (`chmod 775`)
- The *contents* of the images folder to read & write for owner and group, read only for all others (`chmod 664`)
- The images folder itself to full permissions for the owner and group, read and execute only for all others (`chmod 775`)

Finally, set the owner and group for mt-static and all its contents to www, the web server user used by Mac OS X's web services. (`chown -R www:www mt-static/`) Now that we have mt-static taken care of, let's go set up the meat of Movable Type, the cgi directory.

Change directory to /Library/WebServer/CGI-Executables, the default CGI directory for Mac OS X Server. Create a directory called *mt* and change directory into that. (Note: you can just dump all the CGI files into /Library/WebServer/CGI-Executables, but I like to give them their own directory. It makes it easier to keep track of what CGI files are attached to what, and reduces the chance of a random *other* CGi file overwriting your Movable Type CGI files.)

Now, copy all the Movable Type files, (that you didn't copy into mt-static) into this mt directory. (Those of you who are more observant will note that we are doing things out of order of the Movable Type installation instructions. This doesn't make a difference in the end. I happen to prefer uploading, then configuring. If you like to configure, then upload, that works too.)

Change the permissions of all the files ending in .cgi in the mt directory to full permissions for the owner, read and execute only for everyone else. (`chmod 755 *.cgi`) Since we are going to use the BerkeleyDB, we have to create a directory for those files. Create a directory in mt called *db*, and set the permissions to full access for everyone. (`chmod 777 db`) Now, lets create the directory for our first weblog files. Again, by partitioning each blog's files into their own directory, we make our lives a lot simpler later on. So, cd back to /Library/WebServer/Documents/ create a directory, and call it *firstblog*. Inside of this directory, create another directory called *archives*. Set the permissions of firstblog and it's contents so that owner and group have full permissions and everyone else only has read and execute. (`chmod -R 775 firstblog`) Then change its ownership to www for the user and the group. (`chown -R www:www firstblog`). Congratulations, the initial installation of Movable Type is finished. Change directory back to /Library/WebServer/CGI-Executables/mt, and we can start configuring Movable Type.

Initial Movable Type Configuration

In the editor of your choice, open the /Library/WebServer/CGI-Executables/mt/mt.cfg file. I use pico with the –w switch to avoid line wrap issues, but you can use any decent text editor, such as BBEdit, TextWrangler, or SubEthaEdit.

The first thing we want to set is the `CGIPath` for Movable Type. This will let Movable Type know where to start looking for its CGI files. This is *not* the physical disk path to them, but the path you'll see as a URL for our setup, it's going to be:

http://fulldnsnameorIPaddressofserver/cgi-bin/mt/

(This is where you run into one of Apple's Apache Oddities. Yes, the CGI directory is called "CGI-Executables". Ignore that, for down that path lies madness. If you look at the Apache configuration file, httpd.conf in /etc/httpd/, you'll see this line:

```
ScriptAlias /cgi-bin/ "/Library/WebServer/CGI-Executables/"
```

Which tells Apache that the CGI-Executables directory is really the root cgi-bin directory. So even though it seems really wrong, we assume /cgi-bin/ to be /Library/WebServer/CGI-Executables/. Confusing? Yep. Nonstandard? Yep. Is it worth redoing the entire Apache setup for this? Probably not.) So your CGIPath line should look like:

```
CGIPath http://fulldnsnameorIPaddressofserver/cgi-bin/mt/
```

Next, we set the path to our db directory. Since this is a physical disk path, not a virtual HTTP path, we use the "real" path to the db directory, namely:

```
/Library/WebServer/CGI-Executables/mt/db
```

Put that in as the path for the DataSource line in mt.cfg, so it looks like this:

```
DataSource /Library/WebServer/CGI-Executables/mt/db
```

The next thing we need to set is the path to our mt-static directory and its files. This is a HTTP path, and can be relative from the root of the web documents directory, so we just set StaticWebPath to:

```
StaticWebPath /mt-static/
```

That's the minimal configuration to get Movable Type working. However, there are a couple of things that I like to set as well. First, for email notifications of things like comments and trackbacks, I set my mail transfer to smtp and then set my mail server. (Note: You can also just use sendmail locally if you want, but I prefer to use SMTP, since that way, if I change my mail server's machine, I don't have to change my Movable Type settings. It gives me a little more flexibility without a lot of work. So I uncommented the MailTransfer and SMTPServer lines, and set SMTPServer to my mail server: (Obviously, you'll want to set this to *your* mail server)

```
MailTransfer smtp
SMTPServer fulldnsname.ofyourmailserver.here
```

That's pretty much it for mt.cfg. So save your changes and closet that file. Don't think that we've fully configured Movable Type by a long shot. If you take the time to read the mt.cfg file thoroughly, there are a lot of ways you can really tweak Movable Type's setup and performance, so that it works the way you want it to.

Now, we start doing the web configuration of Movable Type. First, we're going to want to check for the available Perl modules. To do this, we use the mt-check.cgi file. So go to:

http://yourwebserver/cgi-bin/mt/mt-check.cgi.

(Note: Yes, I know that the Movable Type instructions don't tell you to put the "cgi-bin" directory in the path when you're

running these CGIs. It's been my experience that you have to, so I just do it and move on. You can probably fix this if you want, but I haven't had a real reason to yet.) You should get a page with a list of configuration information, listing the Perl modules that Movable Type could find. By default, Mac OS X Server should have all the requisite modules installed. It won't always have all the optional ones, but you can always install those later. The important thing is that you get the "Movable Type System Check Successful" message at the bottom of your screen. If not, go back and make sure that you installed all the files to their correct places, set your permissions right, and that mt.cfg is pointing to the right places.

The next step is to initialize the system. This is the make – or – break test for your setup. If you get past this step, then you've got your basic installation and config done. Go to:

http://yourwebserver/cgi-bin/mt/mt-load.cgi.

If you set up everything correctly, then you get the System Initialization Complete message, and dire warnings to delete the mt-load.cgi file. I recommend listening to these dire warnings. However, since we've all been careful, this step works great, and we can now log into Movable Type, and set up our first weblog.

Configuring Movable Type's Web Interface

Go to:

http://yourwebserver/cgi-bin/mt/mt.cgi

This is the main URL for administering your Movable Type setup, so you'll want to bookmark it. The initial userID and password you use is *Melody* for the userID, and *Nelson* for the password. Once you've logged in, you'll see the initial Movable Type administration screen.

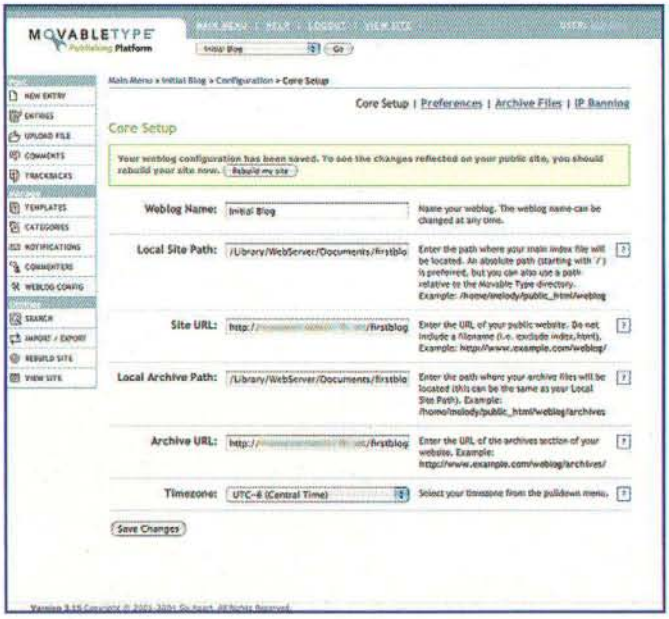
Initial Movable Type Administration Screen

Obviously, you're going to want to create a new weblog author, and give that author full control over your installation, and that default weblog. Once that's done, log out, then log back in as the new author, and edit Melody's permissions so she can't do anything. Unfortunately, you can't delete her, so we'll just cripple her thoroughly. Next we're going to modify that initial weblog so it matches what we want out of it. On the main menu screen, in the "First Weblog" section, click on "configuration".

That will bring you to the "Core Setup" screen, and we can commence changing this to reflect our setup.

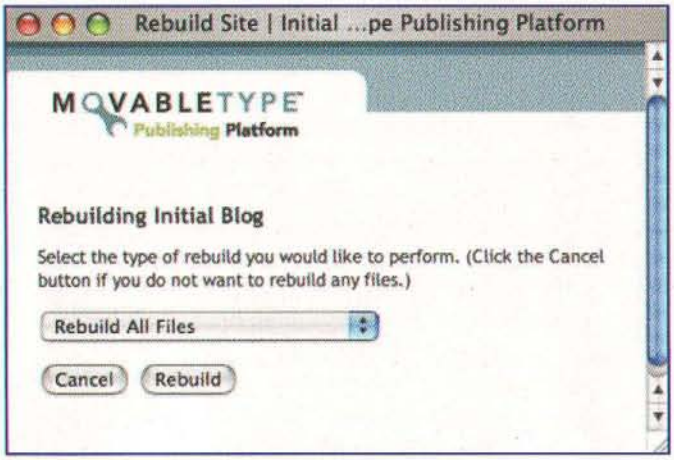
- Change the "Weblog Name:" to whatever you like. This will be the title of the blog for readers, and in the main administration screen.
- Change the "Local Site Path:" to /Library/WebServer/Documents/firstblog, so that all your blog files are in that firstblog directory we created earlier.
- Set the "Site URL:" to http://yourwebserver/firstblog/. This will be the URL for people wanting to read your blog.
- Set the "Local Archive Path:" to /Library/WebServer/Documents/firstblog/archives. This is the physical disk path to the directory in firstblog that will hold the archives of your blog entries.
- Set the "Archive URL:" to http://macservertest.kclife.net/firstblog/archives/. This will be the base URL for all your entry archives.
- Finally, set the "Timezone:" to whatever your local Timezone is.
- Click the "Save Changes" button.

This page will refresh with a new addition, a button that says "Rebuild my site". With Movable Type, the biggest annoyance is the site rebuilding. The current version gives you some ways to avoid this, but you may as well get used to it.



Core Setup screen with rebuild button

Clicking the "Rebuild my site" button will bring up the Rebuild window, shown below. Hit the Rebuild button, and your initial config is now set.

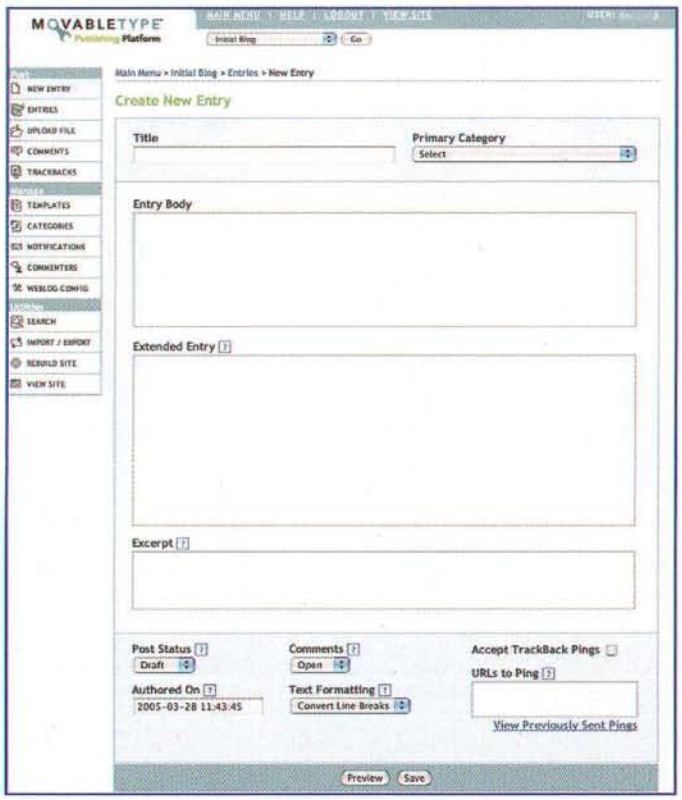


Rebuild Window

Again, there are a ton of configuration options here, but you now have Movable Type basically set up so you can commence to blogging, so, how about we create our first entry?

First Entry

If you look at the column on the left hand side of the "Core Setup" screen, you'll see a link for "New Entry" Click on that, and you'll get the "Create New Entry" screen, seen below.



Movable Type New Entry screen

As you can see, there are quite a few options here, and I'm not going to go over all of them. The online help for Movable Type is excellent, and you should get in the habit of using it as early as possible. Most of the options are pretty self – explanatory. The few I'll point out are that "Entry Body" is what you see below the title on the blog on the main page. "Extended Entry" is what you see when you click on the "continue reading..." links at the bottom of the Entry Body. Use of the Extended Entry is entirely up to you. You don't ever have to use it if you don't want to. The other thing that can catch you off guard is the "Post Status". If you leave it on "Draft", you'll never see your entries on your blog. So, once you're ready to publish your wisdom and wit to the blog, change the "Post Status" to "Publish and hit "Save". Within seconds your entry will be posted, and that's it, you're blogging on Mac OS X Server, even without Tiger.

Conclusion

Obviously, I haven't even begun to touch on all the ways you can customize your blog via custom templates, changing the CSS stylesheets, adding your own stylesheets, etc. There are tons of plugins out there, everything from anti-spam to drop caps for your entries, and you can find all of them on Six Apart's site. If you're going to have your blog on the public Internet, the anti-spam plugins are essentially a necessity, and Movable Type provides one of the better ones, Movable Type-Blacklist, by default. Finally, you don't have to just use Movable Type's "New Entry" screen. There are a multitude of blog editors out there, free and shareware. I personally use ecto, available at http://ecto.kung-foo.tv/. It's not free, but it's one of the best, and has an excellent feature set.

One final note: Blogs aren't just for fun/personal sites. I use them at my "real" job as a way of keeping track of what I'm working on and documenting it as well. There's nothing nicer than having a searchable, easily accessible documentation source that allows for rich text, images, even movies. I find that blogs have made my life as a sysadmin much easier. Simply by giving me a way to keep track of information in a location that is not the ridiculously cluttered top of my desk. So regardless of when Tiger shows up, Mac OS X Server is a great platform for blogging on any level.

About The Author

John Welch <jwelch@bynkii.com> is an IT Staff Member for Kansas City Life Insurance, a Technical Strategist for Provar, (http://www.provar.com/) and the Chief Know-It-All for TackyShirt, (http://www.tackyshirt.com/. He has over fifteen years of experience at making Macs work with other computer systems. John specializes in figuring out ways in which to make the Mac do what nobody thinks it can, showing that the Mac is a superior administrative platform, and teaching others how to use it in interesting, if sometimes frightening ways. He also does things that don't involve computertry on occasion, or at least that's the rumor.

DEV DEPOT

DevDepot has it all!

Get More out of your Mac!

Visit our online store today for special offers and great new products.
www.devdepot.com

KEYSPAN
ONLY $39.99

Digital Media Remote

Control your computer's media programs just like your TV!

For iTunes, Windows Media Player, DVD, CD, and more!

iTripmini
ONLY $39.99

FM Transmitter

Listen in your car!

Designed exclusively for the iPod mini.

NO BATTERIES NEEDED!

iTalk
ONLY $34.99

iPod Voice Recorder

Works as a loud speaker!

Turn your iPod into a world-class voice recorder.

SightLight
FireWire light for iSight

$37.99

AC Adapter
For Powerbook and iBook laptops

$39.99

Keypoint Remote
Multimedia RF remote control

$49.99

Noise Reduction Headphones

$59.99

iMic
ONLY $34.99

USB Audio Interface

A must-have device for people who are serious about high quality audio.

Connect virtually any sound device!

FlashDrive
Cyclops 256 MB USB Flash Drive

$79.99

PowerWave
USB Audio Interface & Amplifier

$89.99

PowerMate
USB Multimedia Controller & Input Device

$36.99

iceKey
Slim USB Keboard

$44.99

TechTool Pro 4.0
for Mac OS X

The ultimate emergency software is now available for OS X!

$89.99

UNIX Utilities
for Mac OS X 3.0

Everything you need to turn your Mac into a UNIX Workstation.

$39.99

Office Applications
for Mac OS X 3.0

Packed with office and productivity apps!

$29.99

Office Applications
for Mac OS X 3.0

Bug reporting and organizing!

* Includes free bug reporter with each application release!

$79.99

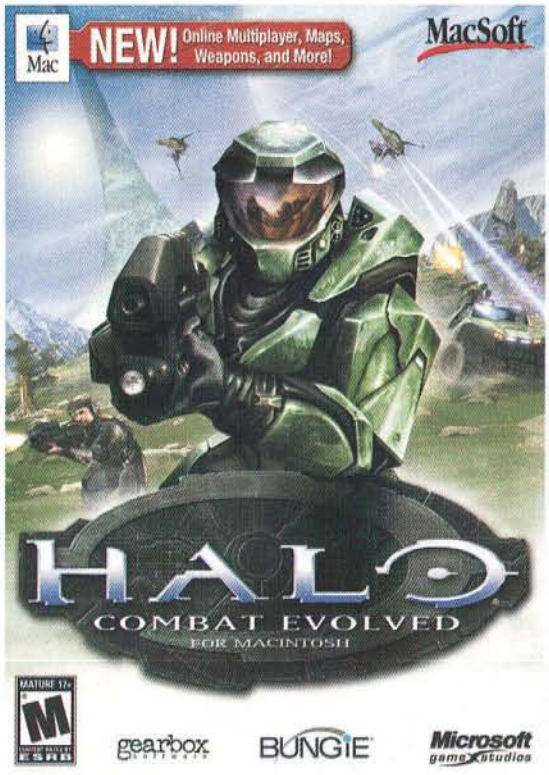
DEV DEPOT

www.devdepot.com

Halo: Combat Evolved

It took forever to get here, but it was worth the wait

This game has a long, and sordid past. It was supposed to be released for Macintosh first, but when Microsoft bought the developer, Bungie, they made Halo: Combat Evolved the premiere title for release with the xBox. If you are interested, a quick Google search will give you more information that you'd care to have on what kind of stink that created. That is neither here nor there, now. The game is out for Macintosh, and while it might be much farther down the road that Mac gamers would have preferred, it's here, and it's amazing.



Like all Bungie games, the action is highly story driven. For Halo, the story centers around the Master Chief, a 26th century genetically enhanced cyborg super soldier, the last of his kind, who is charged with saving humanity. Assisted by an advanced computer AI (sound familiar Marathon fans?), in uncharted space, and on a previously unknown ring world, you, as the Master Chief, set out to save the crew, defeat the alien Covenant, uncover the mysteries of the ring world named Halo, and save the day.

At it's core, Halo is a first person shooter. The controls are similar to many other FPS type games, so it will be easy to jump in and operate within the game's universe. Of course, you can customize the controls to suit your needs. Halo is so much more than many other games of this type, though. You don't just look out of the eyes of your character, nor are you limited to using only certain equipment, or traveling down a specific set of corridors all the time. The levels are huge. The outdoor environments have ample space to explore. Even the underground levels can have multiple ways around. Plus, you get to use just about everything

there. Weapons, vehicles, aircraft, whether it be of human or alien manufacture, are most all usable by you. A real nice feature of the game.

How about the visual environment. The ring world on which the game takes place is not the Larry Niven kind of solar system encompassing ring word. Halo is a much smaller version, orbiting a planet. It's still mighty large, and provides for a great deal of varied terrain to operate on, over, and in. The place is beautiful to look at. The visuals are spectacular. Lots of open areas through which you run missions, as well as buildings, and underground areas to battle in. The quality of the graphics are great. They can be resource hogs, though, so you will need to spend some time tweaking the visual preferences to get the most you can on your system without grinding the game play to a halt. Take a look around, though. Especially up. Seeing the ribbon of the ring in the sky arching over your head is one of the coolest images you'll ever see. Truly impressive.

The sounds in the game are spectacular, too. You can control the detail and quality of it from the preferences, but the higher the quality, and the more you have turned on, the better the game experience will be overall. I'd recommend two things. First, sacrifice some visual quality to keep as much sound on as you can. It's worth it. Second, use a really good set of speakers, or better yet, a good set of head phones, to give you the full auditory experience. It will make all the difference.

Interspersed between missions, levels, and just about any time the game feels like, cut scenes smoothly take over the game, and help advance the story. The scenes themselves are beautifully rendered. They are really nice, fun to watch, and enhance the game.

Online multi-player action is a must in any game these days, and Halo has it, of course. Just go into the Multiplayer menu at the main screen, and you can choose to either join a local LAN game, or get out on the internet, and beat the crud out of other Halo fanatics. For me, it was a sobering experience. I must be getting old, because I went out, and had my guts splattered all over every game I joined. I had fun, though.

As with all things Bungie, a dedicated community has grown up around Halo. Bungie.net is the place to start digging into the community. There are even some novels, published by Del Rey, that delve deeper into the Halo universe. This game has been out on Macintosh for some time now, but it is among the best of what you can get today, so it gets our highest recommendation. Halo for Macintosh is published by MacSoft for Destineer. $50.

By Michael R. Harvey

Advertiser/Product Index

Introducing the bruAPP™

Plug & Play Mac OS X Network Backup System from TOLIS Group

- D2D + D2D2T Backup Appliance
- 250 GB - 1 TB disk stage
- Scalable to support future network growth
- Remote management via graphical and text consoles

- Fully compatible with Xsan environments
- Support for all SCSI and Fibre-Channel tape devices and libraries
- Built in VXA-2 or LTO-2 tape drives in select models
- Client system support for all major operating systems
- BRU™ — 20 years of Backup You Can Trust℠

Mac

One System, One Function, One Solid Performer

5 Minutes From Unpack to First Backup—Literally

Whether you're responsible for a few deskside systems or the latest Hollywood blockbuster, the new bruAPP provides ultra-reliable, easy to use disk-to-disk (D2D) or disk-to-disk-to-tape (D2D2T) network-based backup.

bruAPP complements TOLIS Group's BRU Server for Mac OS X and BRU LE for Mac OS X data backup and restore software products. bruAPP pricing starts at $2,999.

To learn more about the bruAPP and BRU products for Mac OS X, please call 480-505-0488 ext. 252 or visit TOLIS Group on the web at www.tolisgroup.com.

TOLIS Group™

Microsoft
Your potential. Our passion.™

Let the PC world come to you.

Copy and paste. Drag and drop. Switch between Mac and Windows® applications as easily as clicking between documents. The new, faster Microsoft® Virtual PC for Mac Version 7 lets you run PC programs on your Mac like a pro – sans the suit and tie. See it work at www.microsoft.com/mac

Microsoft® Virtual PC for Mac
VERSION 7